Solving the multicommodity flow problem with the analytic center cutting plane method

Babonneau, Frédéric

UNIVERSITÉ DE GENÈVE
Faculté des Sciences Économiques et Sociales
Section des Hautes Études Commerciales

Solving the multicommodity flow problem with the analytic center cutting plane method

Thèse présentée par

Frédéric Babonneau

pour l'obtention du grade de
Docteur ès sciences économiques et sociales
mention gestion d'entreprise

Membres du jury de thèse:
M. Frédéric BONNANS, École Polytechnique et INRIA,
M. Manfred GILLI, Université de Genève,
M. Claude LEMARÉCHAL, INRIA,
M. Olivier SCAILLET, Université de Genève, président du jury,
M. Jean-Philippe VIAL, Université de Genève, directeur de thèse.

Thèse no 603
Genève, 2006

La faculté des sciences économiques et sociales, sur préavis du jury, a autorisé l'impression de la présente thèse, sans entendre, par là, émettre aucune opinion sur les propositions qui s'y trouvent énoncées et qui n'engagent que la responsabilité de leur auteur.

Genève, le 25 avril 2006

Le doyen
Pierre ALLAN

Impression d'après le manuscrit de l'auteur

Preface

This thesis is the result of my researches at the University of Geneva. First of all, I wish express my gratitude to my advisor, Jean-Philippe Vial, for welcoming me at Logilab laboratory, and providing me a very stimulating research environment. His disponibility, his patience and his deep knowledge in the domain of optimization were very precious all along the thesis.

I also want to thank Professor Olivier Scaillet for accepting the presidency of the jury, and Professors Frédéric Bonnans, Manfred Gili and Claude Lemaréchal for being menbers of the jury and evaluating my work.

I am very grateful to Claude Médard for his meticulous reading of the dissertation and for its corrections. I also want to thank the friends and the colleagues of Logilab for creating a pleasant and enjoyable working environment, with a special thank to Laurent Drouet who gave me technical assistance.

The last word is for my wife, parents and friends who have always been encouraging me.

Résumé

La thèse porte sur le développement de méthodes numériques performantes pour la résolution de problèmes de multiflots où plusieurs entités (messages en télécommunications, marchandises ou usagers en transport) entrent en compétition pour l'utilisation d'un réseau à capacité limitée. Pour chaque arc du réseau, le flot qu'il supporte engendre un coût d'utilisation. Ce coût dépend de la somme de messages, usagers ou marchandises circulant sur cet arc. Le but est alors de trouver une solution minimisant la somme des coûts sur l'ensemble des arcs. Les multiflots forment une classe de problèmes importante en recherche opérationnelle. Leur principal défi est la taille des modèles associés aux problèmes rencontrés dans la pratique. Les travaux réalisés dans la cadre de la thèse améliorent les résultats publiés dans la littérature de manière systématique et significative. Ils permettent également de traiter des problèmes dont la taille dépasse de beaucoup ceux rencontrés dans la littérature.

La contribution de la thèse porte principalement sur le développement et la spécialisation d'une mèthode de résolution efficace, ACCPM, et sur la mise en oeuvre d'une stratégie d'ensemble actif permettant de réduire considérablement la taille des problèmes traités.

La méthode ACCPM (Analytic Center Cutting Plane Method) permet de résoudre des problèmes d'optimisation convexes non-diffrentiables en utilisant les concepts de plans coupants et de points intérieurs. Les problèmes couverts par ACCPM sont très généraux et nos expriences révèlent que la méthode est très performante dans son ensemble. La méthode a été spécialisée afin de prendre en compte certaines caractéristiques propres aux problèmes de multiflots: une partie du problème Lagrangien dual différentiable. Ces modifications ont permis d'accélérer significativement les temps de résolution.

La mise en place d'une stratgie d'ensemble actif a également pris une place importante dans l'amélioration des performances de la méthode de résolution. Cette stratégie est motivée par l'observation suivante: dans la plupart des problèmes de transports ou de télécommunications, le nombre d'arcs saturés à l'optimum représente seulement une petite fraction du nombre total d'arcs (10% en général, parfois beaucoup moins). Par conséquent, il est possible de déduire la valeur des variables associées à ces arcs. Si l'ensemble des arcs non saturés à l'optimum était connu à l'avance, il serait alors possible de considérer

un problème équivalent réduit à l'ensemble des arcs saturés. Cet ensemble est appelé ensemble actif, d'oú le nom de la stratégie. Cette réduction rendrait la résolution plus facile. Malheureusement, en pratique, cet ensemble d'arcs n'est pas connu à l'avance. Nous avons donc développé une technique permettant d'estimer l'ensemble actif dynamiquement durant la résolution du problème.

Abstract

The main focus of my PhD research is the implementation of efficient numerical methods to solve the multicommodity flow problems (MCF). MCF mainly arises in the areas of transportation and telecommunications. It consists of routing multiple commodities from a set of supply nodes to a set of demand nodes on a same underlying network. The cost associated with a routing is the sum of costs on the individual arcs. The cost on an individual arc is itself a linear or a nonlinear function of the sum of the commodity flows on that arc. MCF is a challenging optimization problem in operation research domain because of the huge size of instances. It appears that the new proposed approach always improves the results published in the literature in the linear and nonlinear case and makes it possible to solve huge problems.

The main contribution of the thesis is the modification and the specialization of the Analytic Center Cutting Plane Method (ACCPM) and the implementation of an active set strategy that permits to considerably reduce the dimension of the problem.

ACCPM conveniently designates an approach to solve a class of convex nondifferentiable optimization problems. It combines two powerfull optimization techniques : cutting plane methods and interior point algorithms. In the Lagrangian relaxation of the MCF, we point out that the Lagrangian dual objective function has two main components: a piece-wise linear one and smooth one that is the negative Fenchel conjugate of the congestion function. The latter is smooth and can often be computed in closed form. ACCPM has then been specialized to handle explicitly the available smooth component. The new approach considerably improves the performance of the former implementation of ACCPM and makes it possible to solve all the instances.

The implementation of an active set strategy is also proposed to speed up the solution method. This is motivated in linear MCF by our observation that on practical problems, the number of congested arcs in an optimal solution is a small fraction of the total number of arcs in the graph. Consequently, the Lagrangian dual variables associated with these arcs must be null at the optimum. If this (large) set of null optimal dual variables were known in advance, one could perform a partial Lagrangian relaxation restricted to the saturated arcs. This would considerably reduce the dimension of the Lagrangian dual and make it much easier to solve. In practice, the set of

saturated arcs at the optimum is not known. Thus we implemented an active set strategy which dynamically estimates this set during the solution method.

Contents

Chapter 1

Introduction

Contents

1.1 Context of the research

The multicommodity flow problem, in short MCF, is classical in operations research and its numerical solution is a challenging issue (some of real instances have hundreds of millions of variables and hundreds of millions of constraints). Many real-life applications may be formalized under the form of a MCF. It arises in transportation, telecommunication, production, distribution planning, scheduling and management problems. It consists of routing a set of commodities through a given graph from some origins to some destinations at a minimum cost. The most popular application of MCF is the traffic assignment problem. In this problem, the commodities are drivers or vehicles and the graph represents a transportation network with routes, intersections, cities, etc. Here, the goal may be to minimize either the travel time of each driver or the overall travel time in the network.

MCF is considered important enough in the field of optimization to have generated much research on specialized algorithms both in linear programming and in nonlinear optimization. Many primal and dual methods have been proposed in the abundant MCF literature. The most popular primal approaches are either direct [4] or based on Dantzig-Wolfe decomposition [39] in the linear case and, in the nonlinear case, Frank-Wolfe [50] or Simplicial decompositions [69, 120] are more appropriate. Lagrangian relaxation is the most common dual method used in both cases. It yields a Lagrangian dual problem of much smaller dimension which is convex unconstrained but nondifferentiable. Most methods used to solve the Lagrangian dual problem can be classified as cutting plane methods. We can mention Kelley's cutting plane method [73], the bundle method [88], the subgradient method or the analytic center cutting plane method [58], in short ACCPM. In the present dissertation, we revisit ACCPM to improve its performances on multicommodity flow problems.

ACCPM conveniently designates an approach to handle a class of convex optimization problems in which the information pertaining to the function to be minimized and/or to the feasible set takes the form of a linear outer approximation revealed by an oracle. By oracle, we mean a black-box scheme that returns appropriate information on the problem at so-called query points. In convex unconstrained optimization, this information takes the form of a linear support for the epigraph set of the function to be minimized. This class of problems is known as "Nondifferentiable Convex Optimization".

1.2 Contributions of the thesis

The main contributions of the thesis are described below.

First, we point out that the Lagrangian dual objective function of MCF has two main components: a piece-wise linear one and one that is the negative Fenchel conjugate of the congestion function. The latter is smooth and can often be computed in closed form. In a traditional approach with ACCPM [60], the two components are approximated by cutting planes. The intersection of these half-spaces defines a localization set whose analytic center becomes the point where to refine both approximations. Here, we use in the definition of the localization set a direct representation of the epigraph of the smooth component as a fixed constraint. Then we introduce the concept of *second order oracle* to designate a black box that returns appropriate information for the smooth component. The ACCPM structure has permitted the insertion of

such a nonlinear component in the method that is not immediate and evident in others. The scheme is described in [14, 12]

A second main contribution of the thesis is the implementation of an active set strategy to speed up the solution method. This is motivated in linear MCF by our observation that on practical problems, the number of congested arcs in an optimal solution is a small fraction of the total number of arcs in the graph. In other words, for a large majority of arcs, the total flow in the optimal solution is strictly less than the installed capacity. Consequently, the Lagrangian dual variables associated with these arcs must be null at the optimum. If this (large) set of null optimal dual variables were known in advance, one could perform a partial Lagrangian relaxation restricted to the saturated arcs. This would considerably reduce the dimension of the Lagrangian dual and make it much easier to solve. In practice, the set of saturated arcs at the optimum is not known. Thus we implement an active set strategy which dynamically estimates this set. This strategy has been implemented on the linear MCF with success. The results are reported in [13].

We then propose an approximation scheme to extend the idea of active set strategy to the nonlinear MCF. The approximation scheme replaces the nonlinear function near the origin by a linear function. This scheme is motivated by the fact that no Lagrangian dual variables need to be introduced in connection with arcs with a linear cost function. This results in a reduction of the dimension of the Lagrangian dual space and an easier computation of analytic centers. Moreover, when the approximation error is small enough, the optimal solution for the approximated problem is also optimal to the original one. The combination of the approximation scheme and the active set strategy has been successfully implemented for the nonlinear MCF and the results are published in [14].

The present dissertation also proposes a large review of the different MCF formulations found in the literature, particularly in traffic applications. It also reviews and describes shortly the main solution methods used to solve MCF instances.

Finally, an important part time of the thesis has been devoted to the implementation of ACCPM and its extensions [11]. ACCPM has been developed in collaboration with the logilab team, i.e., C. Beltran, O. du Merle, C. Tadonki and J.P. Vial. We can mention as extensions, a proximal term to make the localization set compact, a column elimination strategy to reduce the pool of cutting planes, a strategy to handle inaccurate cutting planes, or a ball constraint that could permit an extension of ACCPM to nonconvex

optimization.

1.3 Outline of the thesis

The thesis is organized into three main parts.

The first part is devoted to MCF. **Chapter** 2 introduces the general of formulation of MCF handled in the thesis and gives some examples of real-life applications. **Chapter** 3 focuses on a particular application of MCF, the traffic assignment problem. We review its different formulations and the main objective functions used in the literature to model the travel times. We introduce the *compound travel time* functions that are relevant in the approximation scheme. In **Chapter** 4, we describe shortly the main solution methods used in the literature to solve MCF problems.

The field of the second part focuses on our solution method. **Chapter** 5 introduces the Lagrangian relaxation of the MCF problem. We point out the two main components of the Lagrangian dual objective. In this chapter, we introduce the active set strategy and the approximation scheme. In **Chapter** 6, we present the analytic center cutting plane method and its enhancements used to solve the Lagrangian dual problem.

The last part includes two chapters in which we apply ACCPM and the active set strategy on MCF instances. **Chapter** 7 is devoted to the linear MCF while in **Chapter** 8, we solve nonlinear MCF. In these chapters, we benchmark our solution method with the most efficient methods used in the literature.

1.4 Notations

For the sake of easier reading, we summarize the main notations used in the present dissertation. In Table 1.1, we report the MCF notations while Table 1.2 gives the notations used in ACCPM.

Notation	Description
x	Vector of individual flows.
y	Vector of total arc flows.
c	Vector of arc capacities.
d	Vector of demands.
r	Vector of linear costs.
$\tilde{g}$	(Delay or congestion) function of the total flow y.
g	Upper extension function of g.
$\tilde{tt}$	Travel time function of the total flow y.
tt	Upper extension travel time function of tt.
δ	Demand function.
N	Node-arc incidence matrix defining the graph $\mathcal{G}$.
$\mathcal{N}$	Set of nodes.
$\mathcal{A}$	Set of arcs.
$\mathcal{K}$	Set of commodities.
$\mathcal{X}$	Set of feasible flows x.
$\mathcal{Y}$	Set of feasible total flows y.

Table 1.1: Notations for MCF.

Notation	Description
f	Objective function.
f_1	Nonsmooth objective function.
f_2	Smooth objective function.
z	Epigraph variable associated to the epigraph of f_1.
ζ	Epigraph variable associated to the epigraph of f_2.
A	Matrix of subgradients of f_1.
E	Binary matrix for cutting planes.
Γ	Vector of right-and-side coefficients in cutting planes.
$\bar{\theta}$	Upper bound for the objective.
$\underline{\theta}$	Lower bound for the objective.
$\mathcal{L}$	Localization set.

Table 1.2: Notations for ACCPM.

Part I

Multicommodity flow problems

Chapter 2

The multicommodity flow problem

Contents

The multicommodity flow problem (MCF) yields formulation of optimization problems that arise in industrial applications such as transportation, telecommunications and logistics. MCF is characterized by a set of commodities to be routed through a network at a minimum cost. In practice, commodities may represent messages in telecommunications, vehicles in transportation or product goods in logistics. Each commodity has to be transported from one or several origin nodes to one or several destination nodes. The cost is a convex or nonconvex function of commodity flows and the individual flows are constrained to be of integer or continuous values. In this thesis the focus is on the continuous and convex case. In this chapter, we introduce the general MCF formulation and we give some examples of immediate and indirect applications.

2.1 Formulation

Given a network represented by a directed graph $\mathcal{G}(\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ denotes the set of arcs, MCF consists of routing different

commodities through this network. We denote the number of arcs n_a, the number of nodes n_n and the number of commodities n_c. Each commodity has to be shipped from a set of supply nodes to a set of demand nodes. Let $\mathcal{K}$ be the set of commodities, the demand vector d^κ for commodity $\kappa \in \mathcal{K}$ is a vector with n_n components constructed as follows:

- $d_i^\kappa < 0$, if i is supply node,
- $d_i^\kappa > 0$, if i is demand node,
- $d_i^\kappa = 0$, otherwise.

In the simpler case, each commodity has only one source node and one destination node.

Let us introduce the following notations for the flow variables. We denote x_a^κ the flow of commodity κ on the arc $a \in \mathcal{A}$, $x^\kappa = \{x_a^\kappa\}_{a\in\mathcal{A}}$ the flow vector for commodity κ and $x = \{x^1, \ldots, x^\kappa\}$ the flow vector for all commodities. The demand constraints are defined by

$$N^\kappa x^\kappa = d^\kappa, \quad \forall \kappa \in \mathcal{K}, \tag{2.1}$$

where N^κ is a subnetwork matrix for commodity κ. In general, we have $N^\kappa = N$, $\forall \kappa \in \mathcal{K}$, where N is the node-arc incidence matrix defining the graph $\mathcal{G}$. The flow of commodity κ may be bounded from above individually by the capacity vector c^κ, so that

$$x_a^\kappa \leq c_a^\kappa, \quad \forall a \in \mathcal{A}, \forall \kappa \in \mathcal{K}. \tag{2.2}$$

When flows are not constrained individually on the arcs, we set the capacity values to $+\infty$. We define the set of feasible flows X such that

$$\mathcal{X} = \{x \geq 0 \mid (2.1) \text{ and } (2.2) \text{ hold}\}. \tag{2.3}$$

The total flow vector, denoted y, is the sum of all commodity flows given by

$$y_a = \sum_{\kappa\in\mathcal{K}} x_a^\kappa, \quad \forall a \in \mathcal{A}. \tag{2.4}$$

Remark 1. *The constraint matrix has a special network structure. The matrices of constraints (2.1)-(2.2) are block angular while constraints (2.4)*

are coupling constraints. The structure is displayed on

$$\begin{bmatrix} N^1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & N^\kappa \\ I & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & I \\ I & \cdots & I \end{bmatrix} \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^\kappa \end{bmatrix} = \begin{bmatrix} d^1 \\ \vdots \\ d^\kappa \\ c^1 \\ \vdots \\ c^\kappa \\ y \end{bmatrix}. \tag{2.5}$$

In view of the above structure, the feasible set (2.3) can be rewriten such as

$$\mathcal{X} = \prod_{\kappa \in \mathcal{K}} \mathcal{X}^\kappa,$$

where

$$\mathcal{X}^\kappa = \{x^\kappa \geq 0 \mid N^\kappa x^\kappa = d^\kappa \text{ and } x^\kappa_a \leq c^\kappa_a,\ \forall a \in \mathcal{A}\}. \tag{2.6}$$

Most solution methods in the literature exploit this special structure of the constraints. They decompose the problem in as many subproblems as the number of commodities, either by linearization of the objective function or by relaxation of the coupling constraints. The main issue is then the coordination of the subproblems. See Chapter 4 for more details.

Any feasible routing solution generates a cost on each arc. The nature of the cost function is specified in Definition 1 and Assumption 1.

Definition 1. *Let $x \in \mathcal{X}$ be a feasible flow and y a total flow such that (2.4) holds. The cost on an arc is assumed to be the sum of a linear function of individual commodity flows and a function of the total flow, denoted $\tilde{g}$. The total cost for the solution (x,y) is given by*

$$\tilde{g}(y) + r^T x,$$

where $r = \{r^1, \ldots, r^\kappa\}$ is a positive cost vector. We denote $r^\kappa = \{r^\kappa_a\}_{a \in \mathcal{A}}$ the cost vector for commodity κ.

Assumption 1. *The function $\tilde{g}$ of the total flow y is assumed to be convex, monotone increasing and separable such that*

$$\tilde{g}(y) = \sum_{a \in \mathcal{A}} \tilde{g}_a(y_a),$$

where

$$\tilde{g}_a : C_a \to \mathbb{R}^+, \quad \forall a \in \mathcal{A},$$

with

$$C_a = [0, c_a], \text{ or } C_a = [0, c_a[.$$

We denote by c_a the mutual capacity on arc a, and c denotes the vector of mutual capacity.

Let us now introduced a relevant definition to formalize the MCF problem.

Definition 2. *Let $\tilde{g}_a$ be a cost function on arc a defined on C_a. We denote by $g_a : \mathbb{R}^+ \to \mathbb{R}^+ \cup \{+\infty\}$ the upper extension function of $\tilde{g}_a$ such that*

$$g_a(y_a) = \begin{cases} \tilde{g}_a(y_a), & \text{if } y_a \in C_a, \\ +\infty, & \text{otherwise.} \end{cases}$$

We defined $g(y) = \sum_{a \in \mathcal{A}} g_a(y_a)$, the sum of the upper extension functions.

In view of the above definitions, the MCF can be formulated as the following optimization problem

$$\min \left\{ g(y) + r^T x \mid y = Mx,\ x \in \mathcal{X} \right\}, \tag{2.7}$$

where the matrix M collects individual flows on the arcs of the network. We assume that problem (2.7) has an optimal, thus finite, solution value. Problem (2.7) is called the *linear multicommodity flow problem* when the cost function $\tilde{g}$ is linear and the *nonlinear multicommodity flow problem* when $\tilde{g}$ is nonlinear. Note that if $\tilde{g}_a(y_a) \equiv 0,\ \forall a$, the function g represents mutual capacity constraints without generating any contribution to the total cost.

Remark 2. *As function g is monotone increasing, the constraints $y = Mx$ may be replaced by inequality constraints. Then (2.7) is equivalent to*

$$\min \left\{ g(y) + r^T x \mid y \leq Mx,\ x \in \mathcal{X} \right\}. \tag{2.8}$$

Remark 3. *An alternative way of introducing mutual capacity in the definition domain of g, is to handle explicit mutual capacity constraints in the model such that*

$$\min \left\{ \tilde{g}(Mx) + r^T x \mid Mx \leq c,\ x \in \mathcal{X} \right\}. \tag{2.9}$$

We easily identify problem (2.9) with (2.7).

Let us now state some assumptions that are implicit in the definition of (2.7).

- The commodities are homogeneous, i.e., every unit of commodity uses one unit of arc capacity (mutual and individual).

- The flow variables x can be fractional. In some applications, the flows must be integer. This type of problem is named *integer multicommodity flow problem*. As pointed out earlier, this thesis does not deal with this type of problem.

Main classes of MCF

The literature on the multicommodity flow problem is abundant. In practice, there exists a wide variety of applications that lead to many variants of MCF. We distinguish two main categories of problems derived from the general formulation (2.7). This classification of MCF is not absolute but most problems fall into one of these categories. (See [44] for a slightly different classification.) In the rest of the dissertation, it is also convenient to refer the reader to these formulations.

The first category groups the linear MCF problems. In this case, the commodities always compete for mutual arc capacities, since, without mutual capacities, the problem is fully separable in commodities and can be solved easily. Each commodity may have multiple supply nodes and demand nodes, and generally the arc cost is a linear function dependent of each commodity flow. The linear MCF can be formulated as follows:

$$\begin{aligned}
\min \quad & r^T x && \text{(2.10a)}\\
& \sum_{\kappa\in\mathcal{K}} x_a^\kappa \le c_a, \quad \forall a \in \mathcal{A}, && \text{(2.10b)}\\
& N^\kappa x^\kappa = d^\kappa, \quad \forall \kappa \in \mathcal{K}, && \text{(2.10c)}\\
& 0 \le x_a^\kappa \le c_a^\kappa, \quad \forall a \in \mathcal{A}, \forall \kappa \in \mathcal{K}. && \text{(2.10d)}
\end{aligned}$$

We easily identify (2.10) with (2.7) if we remember that in (2.7) the mutual capacity constraints are handled in the objective function g. Finding the best route for a single commodity (independently of the other commodities) is then a transshipment problem, in which the number of commodities is usually small to very small with respect to the number of nodes.

In the second category, the arc cost is given by a nonlinear function of total arc flows and each commodity must be shipped from a single origin to a single destination. The commodity flows may be constrained by mutual and/or individual arc capacities. The potential number of commodities may be as large as the square of the number of nodes, a huge number on large networks. But, without individual capacities on the arcs, finding the best route for a single commodity (independently of the other commodities) is a single commodity flow problem. The formulation of problems in this category is given by

$$\begin{aligned} \min \quad & g(y) && \text{(2.11a)} \\ & y_a = \sum_{\kappa \in \mathcal{K}} x_a^\kappa, \quad \forall a \in \mathcal{A}, && \text{(2.11b)} \\ & N^\kappa x^\kappa = d^\kappa, \quad \forall \kappa \in \mathcal{K}, && \text{(2.11c)} \\ & 0 \leq x_a^\kappa \leq c_a^\kappa, \quad \forall a \in \mathcal{A}, \forall \kappa \in \mathcal{K}. && \text{(2.11d)} \end{aligned}$$

The mutual capacity is managed in the domain definition of the objective function. In the following, we will also use the more compact formulation for (2.11)

$$\min \{g(y) \mid y \in \mathcal{Y}\}, \tag{2.12}$$

where Y is the set of feasible total flows

$$\mathcal{Y} = \{y = \sum_{\kappa \in \mathcal{K}} x^\kappa \mid x \in \mathcal{X}\}. \tag{2.13}$$

Remark 4. *When each commodity must be shipped from a single origin to a single destination and the arc cost is a linear function of the total arc flow, the problem can be classified in both categories. In formulation (2.10), mutual capacities are explicitly defined in the model, while in (2.11), they are included in the objective function. In that situation, finding the best route for a single commodity is a simple shortest path problem (when there are no individual capacities).*

2.2 Applications

The multicommodity flow problem arises in a wide variety of important applications. Many transportation, communication, logistic or manufacturing problems can be formulated directly or indirectly as MCF. In this section, we present applications of multicommodity flow problem (MCF) in order to make unaccustomed readers

more familiar with this class of optimization problems. We also desire to point out the importance of MCF in real-life optimization problems. In the previous section, we have introduced mathematical formulations of MCF. Each MCF is characterized by a graph representation and commodity demands to be transported through that graph. Here, we show how different practical problems can be transformed in MCF. The main point is to identify the working graph and its attributes, i.e., arc capacities and arc costs. While the graph representation is direct and evident in routing problems, it is not well-established and often needs transformations in planing and scheduling applications.

2.2.1 Routing and transportation

Many applications consist of routing multiple commodities such as products, messages or drivers. We distinguish two classes of commodities in order to describe two important situations. Either, commodities to be transported are physically different (different manufactured products), or, there is a single kind of commodity (messages or vehicles) but each one is differentiated by a given origin-destination pair of nodes. We give few examples below.

Traffic assignment problem

The most popular application of MCF is the traffic assignment problem [15, 32, 38, 82, 83, 103].

To cope with the problem of traffic congestion, more and more vehicles will be equipped with route guidance systems. Using digital maps, these systems propose a route for each driver from its origin to its destination that have been preliminarily indicated by the drivers. The positions of each drivers are known thanks to a Global Positioning System (GPS) and the route proposals depend on the current traffic congestion. In that case, the traffic problem consists in a purely dynamic model for traffic, where vehicles travel through the transportation network with progressing time.

The main drawback in designing a realistic dynamic model is the difficulty to measure the travel time spent to through an arc. It depends on the number of drivers on that arc at each time period. For this reason, most traffic models leave the time component and consider static flow (constant flow). This approach is quite realistic to describe traffic in rush-hour. In that situation, flow variations between different origins and destinations are reduced over a

long period of time. They are also negligible compared to the travel time due to congestion.

In view of the static model, the traffic assignment problem may be easily formalized as a MCF. The graph represents the transportation network in which nodes are cities or intersections and arcs are route sections. There is a single type of commodity, i.e., vehicles or drivers, and the demands represent numbers of drivers to be routed from their origins to their destinations. In general, the problem is to compute routes for drivers in order to reduce their travel times and avoid congestion on the arcs. The flow of each route section is often upper limited. Chapter 3 is devoted to an extensive presentation and discussion of the traffic assignment problem.

Telecommunication routing problem

The telecommunication routing problem [92, 107] consists of routing messages through a telecommunication network at a minimum cost. The problem is similar to traffic assignment, but in the network, the nodes are origin and destination stations for messages and the arcs represent transmission lines. The demands are the quantity of messages to be routed from an origin to a destination station. Finally, each transmission line[1] has a fixed mutual capacity. Note that in this application, there is only one kind of commodity and no individual capacity on the arcs.

We now give another important problem in telecommunication in which MCF has a central place, the bandwidth allocation in communication network [19, 91, 56]. It consists of assigning capacity expansions in a telecommunication network that make the routing of message demands possible. The objective is to minimize the capacity investment and the cost dues to the routing of messages. Many approaches used to solve this problem yields a subproblem which is a standard MCF and the efficiency of such approaches depends directly on the ability to solve the subproblem.

Other routing problems

- **Computer network problem** [62]
  In this class of applications, nodes represent storage devices, terminals or computer systems, arcs are transmission lines, and commodities represent computer data. The computer network problem proposes to rout data

[1]The capacity of a transmission line is determined by the node transmitters.

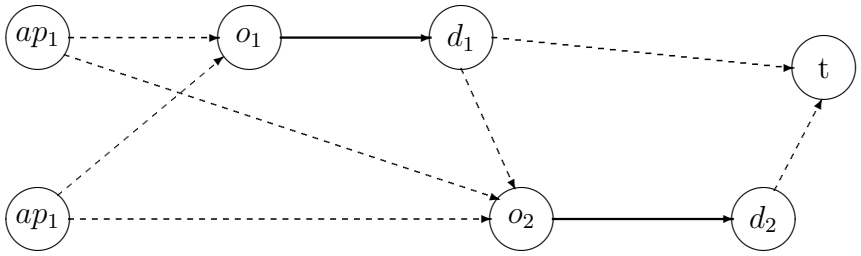

Figure 2.1: Scheduling airplane problem as MCF.

between storage devices, terminals and computers at a minimum cost over mutual capacity constraints[1].

We may also mention the bandwidth allocation problem in computer network [53] that has been described in the context of telecommunications.

- **Distribution problem** [55]
  The distribution problem consists of distributing with trucks or railcars multiple products from plants to retailers using warehouses or railheads. The commodities represent different physical products, and capacity constraints are defined for plants, the warehouses, railheads, and shipping routes.

2.2.2 Scheduling applications

Scheduling is another area of important applications of MCF [9, 3, 16, 8, 18]. To illustrate this class of applications, we present the airline tail assignment problem which can be formulated as an integer MCF [3, 16]. Given a timetable of flight departures and arrivals and a set of airplanes, the objective is to cover the flight legs at a minimum cost assignment by the airplanes. To illustrate this application, we give a small example on Figure 2.1 with two airplanes and two flights. In the formulation, commodities represent airplanes. There are three types of nodes and four types of arcs. The nodes represent airplane origins (ap_1 and ap_2), departures (o_1 and o_2) and arrivals (d_1 and d_2) of flight legs. The arcs represent :

- The first airplane employments between their origins to the flight departures, (ap_1, o_1), (ap_1, o_2), (ap_2, o_1) and (ap_2, o_2).

- The airplane removals from service, (d_1, t) and (d_2, t).
- The flights to be covered between flight departure and flight arrival nodes, (o_1, d_1) and (o_2, d_2).
- Positioning flights between two operatives flights, (d_1, o_2). Here, flight 1 is chronologically before flight 2.

Each arc has a corresponding cost and a unit capacity. Lower bounds on flows are added on certain arcs to ensure that each flight leg is covered by at least one airplane. Note that in this problem, we are interested in 0-1 solutions and thus falls out of the scope of this thesis. Nevertheless, fractional solutions can provide useful bounds for the optimal integer solution. Fractional solutions may also be used in a branch-and bound scheme to compute integer solutions.

2.2.3 Warehousing of seasonal products

The problem of warehousing of seasonal products presented here has been adapted from [4]. A company manufactures several seasonal products with demands varying monthly. The company has the possibility to store a pre-season production in a warehouse of capacity C to supplement peak-season production. The objective is then to determine the production levels for each months that will satisfy the demands and minimize the production and storage costs.

This warehousing problem can be formulate as a multicommodity flow problem defined on an appropriate network. Let first introduce the following notations:

- d_j^k denotes the demand for the products k in month j.
- c_j^k denotes the production capacity of the product k for the jth month.
- r_j^k denotes the production unit cost of the product k for the jth month.
- h_j^k denotes the storage costs of the two products from month j to month $j + 1$.

For the sake of simpler comprehension, we illustrate that problem with a production of two products, $\mathcal{K} = \{1, 2\}$ to be scheduled for the next four months. Figure 2.2 gives the network corresponding to the warehousing problem. The network contains one node for each month (1,2,3 and 4) and

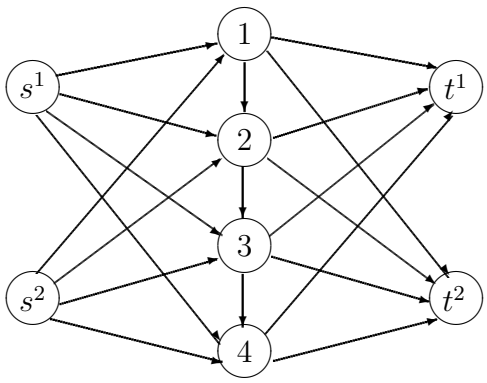

Figure 2.2: Warehousing problem as MCF.

a source (s^1 and s^2) and sink (t^1 and t^2) node for each commodity. The supply and demand of the source and sink nodes is the total demand for the commodity over four months. Only one commodity flows on each of these arcs. We associate a cost r_j^k and a capacity c_j^k with arc (s^k, j). Similarly, the sink node t^k has four incoming arcs; sink arc (j, t^k) has a zero cost and a capacity d_j^k. the remaining arcs are of the form $(j, j+1)$ for $j = 1, 2, 3$. The flow on these arcs represents the units stored from period j to period $j+1$. Each of these storage arcs has a capacity C and a unit cost h_j^k for commodity k. The two commodities share the capacity of this arc.

It is easy to see that each feasible flow in the network is a feasible production and storage schedule for the two products. By optimizing the multicommodity flow, we find the optimum of the warehousing problem.

2.2.4 Optimal deployment of resources

The problem of optimal deployment of resources presented here has been adapted from [4]. Resulting from a natural disaster, a humane organization needs to transport various types of rescue equipments or resources, such as medicine or food, at various locations. The resources are available in different warehouses. Due to technological constraints, it is not possible to satisfy all the demands. Then the organization wants to satisfy the surviving demands minimizing its transportation costs and the "number" of unfulfilled demands.

This problem can be formulated as a multicommodity flow problem. Let us introduce the following notations:

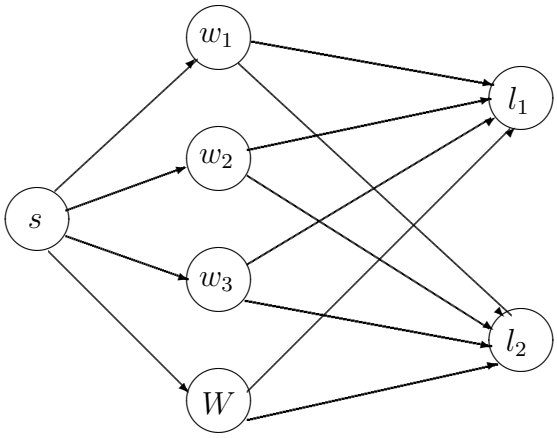

Figure 2.3: Deployment of resources as MCF.

- d_j^k denotes the demand for the resource k at location j.
- a_i^k denotes the amount of resource k available at warehouse i.
- r_{ij}^k denotes the transportation cost of one unit of resource k from warehouse i to location j.
- c_i represents the maximum quantity of all resources that can be transported from warehouse i.
- α_j^k denotes the unit cost of unfulfilled demand of resource k at location j.

For the sake of simplicity, we present a simple situation with three warehouses, two locations and two resources. Figure 2.3 shows the corresponding network. It contains one node for each warehouse (w_1,w_2,w_3), one node per location (l_1 and l_2) and a source node (s). The node W is added to handle unfulfilled demand. The demands of location node l_j for resource k is d_j^k. The supply of the source node for each resource is the sum of all demands. In that formulation, all resources flow on the arcs. We associate on arcs (s, w_i) an individual capacity a_i^k for resource k and a global capacity c_i corresponding to capacities of warehouse i. The associated cost is null. On arcs (w_i, l_j), the flow is not constrained and the cost is resource dependent. For resource k we associate the transportation cost r_{ij}^k. Finally, the remaining arcs deal with the

unfulfilled demands with infinity capacities. The arc (s, W) has a zero cost while the cost α_j^k on arc (W, l_j) depends on the resource.

It is easy to see that any feasible flow in the network is a feasible deployment of resources. Then the optimal deployment is computed by optimization of the multicommodity flow.

Chapter 3

The traffic assignment problem

Contents

This chapter focuses on a particular application of multicommodity flow problem (MCF), i.e., the traffic assignment problem (TAP). TAP is an important transportation application which consists in determining the routes of drivers on a transportation network given their origins and their destinations. In 1952, Wardrop [121] introduced two principles of transportation network utilization leading to the two most commonly used performance criteria. In [35], Dafermos and Sparrow coined the terms *user optimum* and *system optimum* to distinguish the two principles. Previously, they have been discussed in similar terms by Knight [80] and Pigou [106].

The first principle, or user optimum, expresses that the drivers select their routes independently and the travel times of all used routes are less or equal than those which would be experienced by a single driver on any unused route. The user-optimum is also referred in the literature to as a traffic network equilibrium. The second principle, or system optimum, reflects a societal viewpoint. The drivers selects their routes to minimizes the total travel time in the overall network.

In this chapter, we present the different network optimization problems associated with these principles. We also discuss the main differences between them. The last section is dedicated to the analytic form of travel time functions. Since there is no theoretical rules to evaluate the route travel times, many functions have been proposed in the literature. Here, we review the more popular ones used in general routing network context and finally propose alternative functions.

3.1 Problem definition

In the traffic assignment problem (TAP), each commodity (or driver) is characterized by an unique pair of origin and destination nodes. For the sake of simplicity, we consider that commodities compete for capacity on the same underlying network. Recall that N is the node-arc incidence matrix and x represents the flow vector. We define the feasible set of flows as

$$\mathcal{X} = \{x \geq 0 \mid Nx^{\kappa} = d^{\kappa},\ \kappa \in \mathcal{K}\}. \tag{3.1}$$

In this formulation, the capacity constraint on the arc is global and not commodity specific. The vector d^{κ} has only two non-zero components : $-\delta_{\kappa}$ at the origin and δ_{κ} at the destination of the commodity. Thus δ_{κ} is the demand for commodity κ and δ will denote the vector of all demands. We refer the reader to Chapter 2 for general notations.

The problem we consider here is not the more general one in traffic. More realistic, but also more complex models, involve a variety of transportation modes (see [95]).

3.1.1 Travel time function

Given a feasible flow $x \in \mathcal{X}$, we can evaluate the flow performance for each driver by observing the travel time spent for covering the distance from its origin to its destination. This travel time is the sum of the travel times of the used arcs. It is communally assumed that the arc travel time is a direct function of the total flow on that arc. A large flow leads to a high travel time. We recall that the total flow vector y is such that

$$y_a = \sum_{\kappa \in \mathcal{K}} x_a^{\kappa}, \quad \forall a \in \mathcal{A}. \tag{3.2}$$

Assumption 2. *Let $x \in \mathcal{X}$ be a feasible flow and y be the total flow vector given by (3.2). The arc travel time $\tilde{tt}_a$ is a positive, convex and non-decreasing function of the flow on the arc a. The function $\tilde{tt}_a(y_a)$ is defined on the interval $C_a = [0, c_a]$ or $C_a = [0, c_a[$, where c_a is an upper bound for the total flow y_a.*

Assumption 2 implies that the total travel time function is separable. There is no clear argument on the analytic form of the travel time function. The literature proposes many different ones. We review the more popular ones in Section 3.3. Finally, we notice tt_a the upper extension function of $\tilde{tt}_a$ defined by

$$tt_a = \begin{cases} \tilde{tt}_a, & \text{if } y_a \in C_a, \\ +\infty, & \text{if } otherwise. \end{cases}$$

When there is no mutual capacity on a given arc, we get $c_a = +\infty$ and the two functions $\tilde{tt}_a$ and tt_a are equivalent.

Remark 5. *Our assumption that the travel time is a function of the flow may be subject of controversies. In [97], Nesterov suggests that it is more realistic to lead the travel time with the number of drivers present on the arc, i.e., the density. We will discuss this approach in Section 3.3.*

3.1.2 Path-flow reformulation

Up to now, we have described MCF problems with arc flows. We propose to reformulate demand constraints and travel times using path flows between each OD pairs instead of arc flows. This formulation, known as path-flow formulation, is useful in the following.

Let us denote π a path (or a route) on the graph from some origin to some destination. A path is conveniently represented by a Boolean vector on the set of arcs, with $\pi^a = 1$ if and only if the path goes through the arc a. For each commodity $\kappa \in \mathcal{K}$, we denote $\{\pi_j\}_{j \in J_\kappa}$ the set of paths from the supply node to the demand node and δ_κ the demand value. Finally, the flow on path π_j is denoted ξ_j^κ. Using the above notations, the demand constraints in path-flow formulation are defined by

$$\sum_{j \in J_\kappa} \xi_j^\kappa = \delta_\kappa, \quad \forall \kappa \in \mathcal{K}, \tag{3.3}$$

and the total flow on the arcs is

$$y_a = \sum_{\kappa \in \mathcal{K}} \sum_{j \in J_\kappa} \xi_j^\kappa \pi_j^a, \quad \forall a \in \mathcal{A}. \tag{3.4}$$

A commodity κ routed on a path $j \in J_\kappa$ generates the travel cost

$$c_j^\kappa = \sum_{a \in \mathcal{A}} tt_a(y_a)\pi_j^a. \tag{3.5}$$

In a path-flow formulation, the demand constraints (3.3) have a simpler structure than in the arc-flow. There is only one demand constraint for each commodity. Unfortunately, the number of variables is huge even on small networks. Besides, it growths exponentially with the size of the network making that definition impractical.

3.1.3 Fixed and elastic demand

Most of the time the demands δ_κ are assumed to be fixed and known. In some applications, this assumption must be relaxed to account the fact that long travel times adversely affect the demand. In other words, the demands are elastic, which we formalize in the following assumption.

Assumption 3. *The demand δ_κ for commodity κ only depends on the shortest travel time from the supply node to the demand node. The demand function $\delta_\kappa(s)$, where s is travel time along the shortest path and*

$$\delta_\kappa : \mathbb{R}^+ \to \mathbb{R}^+, \quad \kappa \in \mathcal{K},$$

is continuously differentiable, non-negative, upper bounded, and strictly decreasing.

The particular case of a constant function corresponds to an inelastic demand, i.e., fixed demand.

These assumptions on the demand function are standard [71]. For the sake of simpler notation, we note in the following the set of feasible flows $\mathcal{X}(\delta)$ when the demand is elastic.

3.2 Traffic assignment models

In this section, we present different TAP formulations as optimization problems based on the performance criteria stated by Wardrop's principles [121]. Let us recall these principles.

- **First principle**: The journey times of all used routes are less or equal than those which would be experienced by a single vehicle on any unused route.

- **Second principle**: The average journey time is minimal.

The first principle leads to user optimum models and the second one gives system optimum models. In the last part of this section, we show differences between the two approaches.

3.2.1 User optimum models

The Wardrop first principle is also known as *user equilibrium*. In that situation, the users are considered selfish. They select their own route unilaterally, in their own self-interest. The selfish behavior may result in a stable state, named user equilibrium. The basic idea is that, at equilibrium, no user can get a quicker travel time by changing unilaterally his route. In other words, each user pursues individually to minimize his/her own travel time. An equilibrium is characterized by the following equilibrium conditions. For all commodity $\kappa \in \mathcal{K}$ and all paths $j \in J_\kappa$, we have

$$\xi_j^\kappa > 0 \quad \Rightarrow \quad c_j^\kappa = \min_{p \in J_\kappa} c_p^\kappa, \tag{3.6a}$$

$$\xi_j^\kappa = 0 \quad \Rightarrow \quad c_j^\kappa \geq \min_{p \in J_\kappa} c_p^\kappa. \tag{3.6b}$$

The fixed demand model

Let us consider here TAP with inelastic demands δ_κ for each commodity $\kappa \in \mathcal{K}$. In 1956, Beckmann et al. [17] were the first to formulate Wardrop's first principle mathematically, in the case of separable cost functions. They established the equivalence between these equilibrium conditions and the KKT conditions of a convex mathematical programming problem. Then, Dafermos [32] formulates this program under the assumptions that the travel time function tt is integrable and its Jacobian is positive semidefinite for all feasible flows. Let us introduce the definition:

Definition 3. *Let $\tilde{tt}_a(y_a)$ be an integrable travel time function on the arc $a \in \mathcal{A}$ satisfying Assumption 2 and $tt_a(y_a)$ be its upper extension function. The associated delay or congestion function is given by*

$$g_a(y_a) = \int_0^{y_a} tt_a(x)dx.$$

We note

$$g(y) = \sum_{a \in \mathcal{A}} g_a(y_a).$$

Then the equilibrium conditions (3.6) are equivalent to solving the optimization problem:

$$\min_{x,y} \left\{ g(y) \mid y = \sum_{\kappa \in \mathcal{K}} x^{\kappa},\ x \in \mathcal{X} \right\}. \tag{3.7}$$

However, the integration assumption is too restrictive in some applications. In the case of non-integrable functions, the partial derivatives become asymmetric and the problem is known as the *asymmetric traffic assignment problem*. Wardrop's first principle can be formulated therefore as a variational inequality problem [33, 116]. Theorem 1 shows that the solution of a variational inequality problem satisfies the equilibrium conditions (3.6).

Theorem 1. *(Theorem 4.5 of [95])*
A vector y^* is an equilibrium pattern if and only if it satisfies the variational inequality problem

$$tt(y^*).(y - y^*) \geq 0, \quad \forall y \in \mathcal{Y}, \tag{3.8}$$

where

$$\mathcal{Y} = \{ y = \sum_{\kappa \in \mathcal{K}} x^{\kappa} \mid x \in \mathcal{X} \}.$$

Under the assumption of continuity for function tt the equilibrium pattern y^* exists [95]. Variational inequality formulation (3.8) allows to consider traffic assignment problems with more general extensions such that link interactions, multiple classes of users or multiples modes of transportation. User optimum can be also formulated as a nonlinear complementarity problem [1, 2] or as a nonconvex mathematical programming problem [64].

The model with elastic demand

In that part, we consider that the demand δ_κ on the OD-pair (i, j) is elastic and only depends on the shortest travel time from the origin i to the destination j (see Assumption 3). The model developed here is due to Dafermos [34].

Definition 4. *Let δ_κ be the demand functions for commodity $\kappa \in \mathcal{K}$ satisfying Assumption 3. The demand disutility function for the commodity κ is*

$$h_\kappa(\delta_\kappa) = -\int_0^{\delta_\kappa} \omega_\kappa(s) ds, \tag{3.9}$$

where $\omega_\kappa(s) = \delta_\kappa^{-1}(s)$ is the inverse function of δ_κ. We note

$$h(\delta) = \sum_{\kappa \in \mathcal{K}} h_\kappa(\delta_\kappa).$$

Our assumptions imply that the demand disutility function is convex and differentiable by Definition 4. Then, in the case of separable and integrable travel time functions, the equilibrium conditions (3.6) can be obtain as the solution of the optimization problem (see [103]):

$$\min_{x,y,\delta} \left\{ g(y) + h(\delta) \mid y = \sum_{\kappa \in \mathcal{K}} x^{\kappa},\ x \in \mathcal{X}(\delta) \right\}. \tag{3.10}$$

As for the fixed demand situation, Wardrop's first principle can be obtained by solving a variational inequality problem; see Theorem 2.

Theorem 2. *(Theorem 4.1 of [95])*
A vector (y^*, δ^*) *is an equilibrium pattern if and only if it satisfies the variational inequality problem*

$$tt(y^*).(y - y^*) - \omega(\delta^*).(\delta - \delta^*) \geq 0, \quad \forall (y, \delta) \in \mathcal{Y}. \tag{3.11}$$

where

$$\mathcal{Y} = \{(y, \delta) \mid y = \sum_{\kappa \in \mathcal{K}} x^{\kappa},\ x \in \mathcal{X}(\delta)\}.$$

Recent studies [104, 105] focus in a more general traffic equilibrium framework, where demand function is possibly non-separable or non-invertible.

3.2.2 System optimum models

The Wardrop second principle corresponds to a situation in which the total travel time of the system is minimized. The users select their routes from a societal point of view. This Wardrop principle is known as *system optimum*. Under the assumption that the travel time function is separable, monotone, and convex, system optimum can also be formulated as a convex mathematical programming problem defined by

$$\min_{x,y} \left\{ \sum_{a \in \mathcal{A}} tt_a(y_a) y_a \mid y = \sum_{\kappa \in \mathcal{K}} x^{\kappa},\ x \in \mathcal{X} \right\}. \tag{3.12}$$

In problem (3.12), demands δ_κ for each commodity $\kappa \in \mathcal{K}$ are fixed and known. Let $\omega_\kappa(s) = \delta_\kappa^{-1}(s)$ be the inverse function of the elastic demand function δ_κ as in the previous part. Then, under the assumption that ω is a

concave function, system optimum with elastic demands is the solution of the optimization problem:

$$\min_{x,y,\delta} \left\{ \sum_{a \in \mathcal{A}} tt_a(y_a) y_a - \sum_{\kappa \in \mathcal{K}} \omega_\kappa(\delta_\kappa)\delta_\kappa \mid y = \sum_{\kappa \in \mathcal{K}} x^\kappa,\ x \in \mathcal{X}(\delta) \right\}. \tag{3.13}$$

The literature related to system optimum is not as extensive as for the user optimum, partly because the second principle may lead to solutions that are unfair to some users. We discuss of such a behavior in the next paragraph.

3.2.3 User vs. system optima

Le us now discuss relations between user optimum and system optimum. It is well-known that the two notions are not equivalent. A user equilibrium satisfies all the users but it does not necessarily minimize the total travel time through the network. In the system optimum, unacceptable long paths may be assigned to drivers in order to use shorter paths for many other drivers.

Braess's paradox

This situation is illustrated by the so-called Braess paradox [23]. In that paper, the author presents a simple example in which adding an arc in the network yields worse users travel times at user optimum.

Let a problem be characterized by the network defined on Figure 3.1 with the arc set $\mathcal{A} = \{a, b, c, d\}$ and node set $\mathcal{N} = \{1, 2, 3, 4\}$. There is one commodity to be routed from node 1 to destination node 4 with a fixed demand of 6. Two paths are available for routing the commodity, $j_1 = (a, c)$ and $j_2 = (b, d)$. The travel time functions are defined such that:

$$\begin{aligned} tt_i(y_i) &= 10y_i, \quad i = \{a, d\}, \\ tt_i(y_i) &= y_i + 50, \quad i = \{b, c\}. \end{aligned}$$

It is easy to verify that user and system optimum are obtained when affecting 3 units on path j_1 and 3 units on path j_2 such that:

$$y_i^* = 3, \quad i = \{a, b, c, d\}.$$

The associated path travel times are

$$c_{j_1}^* = c_{j_2}^* = 83.$$

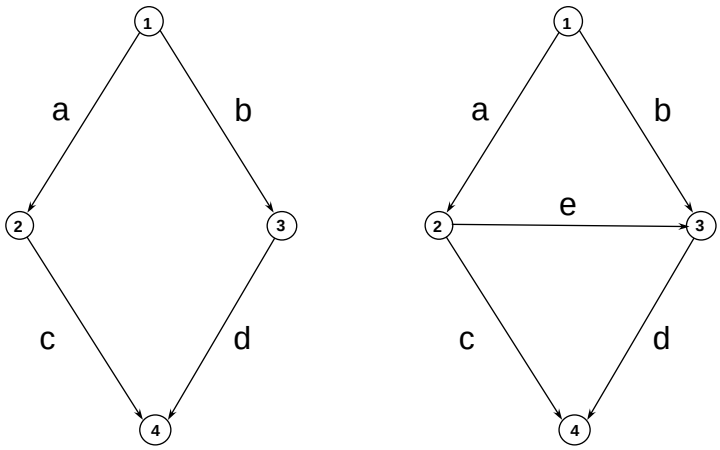

Figure 3.1: The Braess network example.

Assume now that a new arc e from node 2 to node 3 is added in the network (see Figure 3.1). The travel time function of this new arc is $tt_e(y_e) = 10 + y_e$. That results in a new available path $j_3 = (a, e, d)$. There is now three available paths. It is easy to verify that the system optimum solution does not change. Since switching from path j_1 to new path j_3, a commodity may reduce its travel time to 81, the last solution is not more an equilibrium. The user equilibrium pattern is obtained when routing 2 units of flows on each path such that:

$$y_i^* = 2,\ i = \{b, c, e\}, \quad \text{and} \quad y_i^* = 4,\ i = \{a, d\},$$

with path travel times

$$c_{j_1}^* = c_{j_2}^* = c_{j_3}^* = 92.$$

Then, adding a new arc in the network leads to a worse user equilibrium travel time but it can not increase the system optimum solution. This small example confirms that user optimum and system optimum are not equivalent.

Price of anarchy

The price of anarchy, a notion introduced in [108], measures the user optimum inefficiency in term of total travel time. In other words, it measures how bad is user selfish with respect to the system optimum solution? The price of

anarchy is the coefficient defined such as

$$\text{Price of anarchy} = \frac{\text{Total travel time at user equilibrium}}{\text{Total travel time in system optimum}}.$$

Recent works have evaluated the price of anarchy. In [109], the authors showed that for uncapacitated problem the total travel time associated to the user optimum is at most two times the minimum travel time. This ratio falls to 4/3 for linear travel time functions. The main result is due to Roughgarden. In [108], he proved in the case of nondecreasing and differentiable travel time functions that the worse-case of user optimum inefficiency is independent of the network topology. For any travel time function tt, the total travel time produced by user optimum is at most $\alpha(tt)$ time the system optimum in an uncapacitated network. The parameter $\alpha(tt)$ is only function dependent. In [31], the authors extended the above result for capacitated networks and more general travel time functions. They proved that the coefficient α found for uncapacitated network is the same with capacity constraints. The parameter α is also capacity independent.

Constrained system optimum

In that subsection, we present an approach introduced in [70] which deals with both notions, user and system optima. The authors propose to compute the system optimum in which user constraints are added. Additive constraints guarantees that user travel times of the system optimum are not so far from user travel times obtained with the user optimum model.

Assume that normal lengths τ_j^κ are given for each possible path $j \in J_\kappa$ and for all commodity $\kappa \in \mathcal{K}$. Generally, normal lengths are chosen to be the user optimum travel times. They are computed in advance and are fixed in the model. Then the approach proposes that longest paths with respect to the normal lengths are declared unfair and are excluded from the feasible set of paths. Let $\phi > 1$ be a tolerance factor, the feasible set of paths for commodity κ, $\bar{J}_\kappa \subset J_\kappa$, is defined as

$$\bar{J}_\kappa = \{j \in J_\kappa \mid \tau_j^\kappa < \phi\tau^\kappa\},$$

where

$$\tau^\kappa = \min_{i \in J_\kappa} \tau_i^\kappa.$$

Combining system optimum model (3.12) and the above definition, the constrained system optimum, called CSO$^\phi$, is the solution of the optimization

problem:

$$\min \quad \sum_{a\in\mathcal{A}} tt_a(y_a)y_a \tag{3.14a}$$

$$\sum_{j\in\bar{J}_\kappa} \xi_j^\kappa = \delta_\kappa, \quad \forall\kappa\in\mathcal{K}, \tag{3.14b}$$

$$\sum_{\kappa\in\mathcal{K}}\sum_{j\in\bar{J}_\kappa} \xi_j^\kappa \pi_j^a = y_a, \quad \forall a\in\mathcal{A}. \tag{3.14c}$$

Note that in that formulation, the summation is taken in j over the set $\bar{J}_\kappa \subset J_\kappa$ In [70], the authors show that solving (3.14) ensures total travel times of constrained and unconstrained system optimum to be still close. Moreover, the proposed approach improves the fairness of user travel times compared to the unconstrained system optimum travel times. The main drawback is that when applying decomposition methods, the subproblems are NP-hard. In that situation, the subproblem is a sum of constrained shortest path problems.

3.3 Travel time and delay functions

In this section, we present the main travel time and delay functions used in routing applications and more precisely in transportation and telecommunication problems. Since it does not exist a standard way of expressing the travel time in function of the flow, the analytic form of such functions is usually a subject of dispute. Nevertheless, the *Kleinrock* function and the *BPR* (Bureau of Public Roads) function appear to be widely used, respectively, in telecommunications and transportation. Many classes of functions have also been suggested by Branston in [24]. In the last subsection, we discuss some observations that lead to alternative travel time functions, called compound functions. For the sake of simpler notation, we write in the rest of the section g for g_a, tt for tt_a, y for y_a, c for c_a, etc.

3.3.1 Linear delay function

Linear delay functions including upper bound on flow are probably the simplest and the mostly used function in classical MCF, see [13, 49, 84, 93]. The travel time is a constant given by

$$tt(y) = t, \quad y\in[0,c], \tag{3.15}$$

where constant $t \geq 0$. The associated delay function is the linear function

$$g(y) = ty, \quad y \in [0, c]. \tag{3.16}$$

We identify two special cases of linear functions:

- The uncapacitated linear function. In that situation, we have $c = +\infty$ and the traffic assignment problem yields to a sum of independent shortest path problems.
- The indicator function where $t = 0$. This function is useful to compute feasible flows with respect to capacity links.

3.3.2 Kleinrock function

The most frequently used function in the telecommunication domain, see [22, 60, 79, 102], is the Kleinrock function [79]. This function has a vertical asymptote when approaching the arc capacity. Other asymptotic travel time functions are presented in [36, 37]. The travel time on an arc is computed with

$$tt(y) = \frac{c}{(c-y)^2}, \quad y \in [0, c[, \tag{3.17}$$

where c is the capacity on the arc. The function has a vertical asymptote at $y = c$. The associated delay function is

$$g(y) = \frac{y}{c-y}, \quad y \in [0, c[. \tag{3.18}$$

Kleinrock travel time and congestion functions are plotted on Figure 3.2.

This kind of functions includes implicitly capacities in the objective function. It seems to be a useful alternative to adding explicit capacity constraints in the problem formulation. The vertical asymptote also prevents arcs to be saturated and generally leads to a better spread of the flows in the network. Nevertheless in some applications, cost functions with a vertical asymptote lead to unrealistic high travel times when flow approaches the capacity. Furthermore, as mentioned in [83], vertical asymptotes are known to induce numerical difficulties on many optimization schemes.

3.3.3 BPR function

Most of travel time functions used in transportation problems are polynomial functions. The wildly used one, see [15, 28, 38, 83], is the standard function

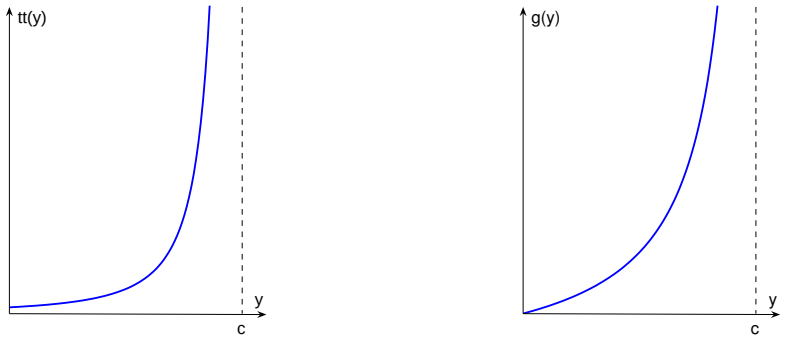

Figure 3.2: Kleinrock travel time and delay functions.

introduced by the U. S. Bureau of Public Roads [68], known as *BPR* function. Its simplicity is certainly the main reason of its success. For each arc, the travel time is provided by

$$tt(y) = r\left(1 + \alpha\left(\frac{y}{c}\right)^{\beta}\right), \quad y \in [0, +\infty[. \tag{3.19}$$

The associated delay function is

$$g(y) = ry\left(1 + \frac{\alpha}{\beta+1}\left(\frac{y}{c}\right)^{\beta}\right), \quad y \in [0, +\infty[. \tag{3.20}$$

We plot the BPR travel time and delay functions on Figure 3.3. In general, the parameter α is small and $\beta > 1$ does not exceed 5. When the flow y is less than c, the second term under the parenthesis in (3.19) is negligible. Thus $g(y) \approx ry$: the parameter r is called *free-flow travel time* and it can be interpreted as a fixed travel time on a congestion-free arc. For larger values of y the nonlinear contribution to congestion increases. The threshold value c for the flow y is usually named the *practical capacity* of the arc, beyond which congestion becomes effective. The standard values, suggested in [114], are $\alpha = 0.15$ and $\beta = 4$. In [43], the authors propose to set $\alpha = 1$ and $\beta = 10$. They validate the study with results of operational models that test freeway and arterial sections. Other studies propose to set $\alpha = 0.2$ or $\alpha = 0.05$ and $\beta = 10$. In some applications, the parameters α and β are arc-dependent.

The main drawback of the BPR function is associated to high values of β. In that situation travel time is generally overestimated when flow is below

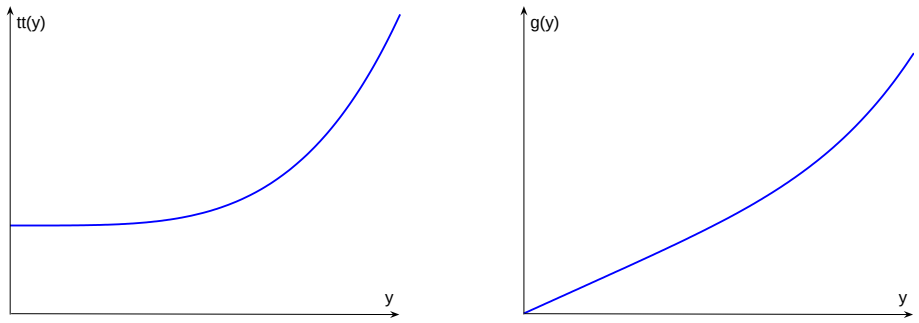

Figure 3.3: BPR travel time and delay functions.

the capacity. On the other hand, when flow exceeds capacity, high values of β may slow down the convergence to user equilibrium of solution methods and cause numerical difficulties.

The other weakness of the BPR function is that improbable high flows may be assigned on the arcs of the network. In practice this assumption seems unrealistic. We are going to explain the reasons later in a next subsection. In [83], the authors propose to limit the definition domain of the flow introducing explicitly capacity constraints in the model. The upper bounds on the flows are constructed such that $y \leq Kc$, where $K > 0$ is a capacity constant.

3.3.4 Conical function

To provide credible alternative to BPR functions, conical travel time functions were introduced in [117]. They were developed and calibrated with information directly based on real transportation applications. The arc travel time is given by

$$tt(y) = t_0 \left(2 + \sqrt{\alpha^2(1 - y/c)^2 + \beta^2} - \alpha(1 - y/c) - \beta\right), \quad y \in [0, +\infty[,$$

with

$$\beta = \frac{2\alpha - 2}{2\alpha - 1}.$$

In that formulation, α is a parameter for congestion effect; c represents the critical flow and t_0 is the free flow time. The delay function associated

to the conical travel time function has the following complicated analytical expression.

$$g(y) = t_0 y(2-\alpha-\beta) - \frac{ct_0}{2\alpha}\left[\alpha(1-\frac{y}{c})\sqrt{\alpha^2(1-\frac{y}{c})^2+\beta^2}\right.$$

$$\left.+\beta^2 \ln\left(\alpha(1-\frac{y}{c}) + \sqrt{\alpha^2(1-\frac{y}{c})^2+\beta^2}\right)\right] + \frac{t_0\alpha y^2}{2c} + ct, \quad y \in [0,+\infty[.$$

We plot on Figure 3.4 the functions $tt(y)$ and $g(y)$.

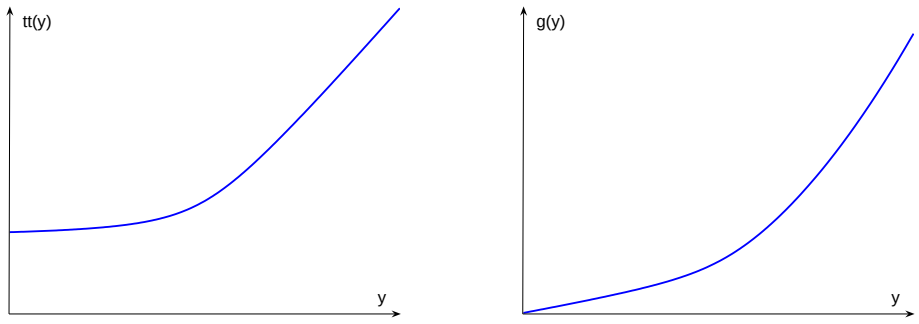

Figure 3.4: Conical functions.

This conical function has the advantage to partly overcome drawbacks of the BPR function. When flow is below the capacity, conical function is very similar to BPR function. Otherwise when flow exceeds capacity, travel time can not become too high. That makes solution methods easier to convergence to the user equilibrium. Nevertheless, this function always yields a travel time at capacity of twice the free travel time — something which may not always be desirable.

3.3.5 Davidson's function

Davidson's function has been first introduced in [40]. Then, calibrations and improvements of the proposed function have been elaborated in [5, 41, 119]. The travel time function is based on queuing theory and has the form

$$tt(y) = t_0\left(1 + J_D\frac{y/c}{1-y/c}\right), \quad y \in [0,c[, \tag{3.21}$$

where t_0 represents the zero-flow travel time; c is the capacity; and J_D is a delay parameter (or $1 - J_D$ is a quality of service parameter). The analytic expression of g is given by

$$g(y) = t_0 y - t_0 y J_D - t_0 c J_D \ln(c - y) + ct, \quad y \in [0, c[.$$

The above functions are plotted in Figure 3.5.

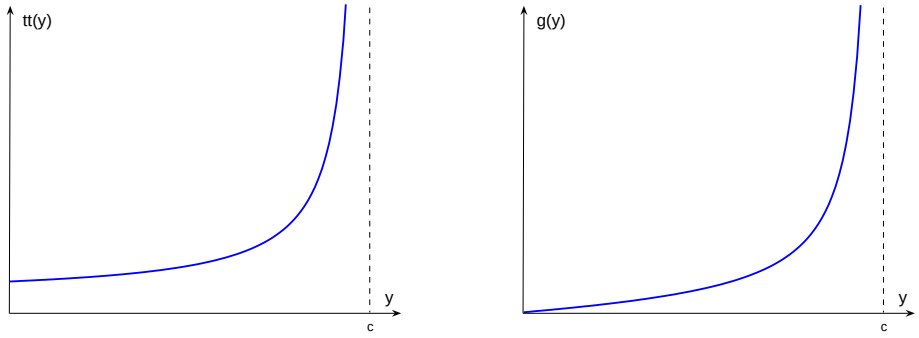

Figure 3.5: Davidson's functions.

As for the Kleinrock function, this model may return infinite travel time when flow approaches the capacity. That may be unrealistic in practical applications.

3.3.6 Discussion on travel time functions

In this subsection, we discuss some observations that may lead to alternative travel time functions. We partly use observations made by Nesterov in [97].

Compound travel time functions

In the literature, most travel time functions are calibrated to limit or neutralize the influence of a low flow on the computed travel time. Assuming that a small flow on a route is induced by small number of drivers on that route, it seems quite natural to assign the free travel time for all drivers on that route. In that situation the route is non-congested. Then an alternative relevant way to describe this property is to model travel time with a *compound travel time*

function. This function computes travel time such that

$$tt(y) = \max(t_0, \gamma\tilde{t}t(y)), \tag{3.22}$$

where t_0 is the free flow time, γ is a parameter and $\tilde{t}t$ is a given travel time function. We assume that $t_0 > \gamma\tilde{t}t(0)$. Let y^c be the intersection between t_0 and $\gamma\tilde{t}t(y)$, the route remains non-congested for flow below y^c. The associated delay function has the form

$$g(y) = \max(t_0 y, \gamma\tilde{g}(y) + ct), \tag{3.23}$$

where $\tilde{g}$ is the delay function associated to the travel time function $\tilde{t}t$ and ct is a constant ensuring that the switching point in the equation (3.22) is the same as in (3.23). Let us provide in Table 3.1 some examples of compound travel time functions. We also give the associated *compound congestion functions*. Note

	Travel time function	Congestion function
Kleinrock case	$\max\left(t_0, \gamma\frac{c}{(c-y)^2}\right)$	$\max\left(t_0 y, \gamma\frac{y}{c-y} + ct\right)$
BPR case	$\max\left(t_0, \gamma r\left(1 + \alpha\left(\frac{y}{c}\right)^\beta\right)\right)$	$\max\left(t_0 y, \gamma r y\left(1 + \frac{\alpha}{\beta+1}\left(\frac{y}{c}\right)^\beta\right) + ct\right)$

Table 3.1: Compound functions.

that it is also possible to develop the approach of compound functions from a delay function viewpoint. Considering that delay function has to be linear to a given threshold of flow, we first define compound congestion function as $g(y) = \max(t_0 y, \gamma\tilde{g}(y))$. Then we deduce the travel time function. The two above approaches are mostly similar.

A new approach

The assumption that the travel time is a function of the flow may be subject of controversies. Here, we describe an alternative approach for travel time function proposed by Nesterov [97]. Let us first report observations which link flow, speed and travel time. See [97] for a slightly different viewpoint. Generally a highly congested route induces a low flow. That results in a very large travel time. On the other hand, when a flow on a route is large, the speed is typically large and the resulted travel time is then small. These two intuitions refute the assumption that travel time is an increasing function of the flow. Moreover a low flow is due typically either to a high density of

drivers on the route (i.e., congested route) or to a very low density. Thus it seems more relevant to link speed, flow and density using the fundamental Little law,

$$flow = speed \times density. \tag{3.24}$$

Let rewrite (3.24) such that

$$travel\ time = density/flow. \tag{3.25}$$

In view of the above equation, travel time is an increasing function of the flow only if a flow augmentation is compensated by a sufficient expansion of density. Assuming that route density is bounded above, that is realistic, then it is quite natural to affect an explicit capacity on flow. In [97], Nesterov mentions that possible augmentation of density is too limited to model in a relevant way travel time with only flow information. He proposes to model travel time with the model

$$tt(d) = \max(t_0, d/c), \tag{3.26}$$

where t_0 is the free flow time and c represents the flow capacity on the arc. The variable d is the number of drivers present on the arc. Note that (3.26) is density dependent, the flow interferes only with its capacity. Thus it seems difficult to handle conservation of flows in a mathematical problem using this model. The author provides new theory based on stable traffic equilibria which is strongly related to Wardrop's first principle. Moreover he shows that instead of using explicit travel time functions, equilibrium is reachable only by introducing in the model lower bounds on travel times and upper bounds on flows. These results are surprising but quite promising (Finding bounds on flows and travel times seems easier than calibrating any travel time functions). Finally, this approach points out that it is more realistic to reach user equilibrium using linear travel time functions instead of using nonlinear ones.

3.4 Summary

In this chapter, we have presented two formulations of TAP, the user and system optima. The user optimum models a selfish behavior of drivers while the system optimum reflects a societal point of view. We also mentioned that the demand may be elastic and thus depends on the amount of drivers on the network. Finally, in the last section we gave the main travel time functions

used in the literature. In this thesis, we focus our numerical experiments on the user optimum with different travel time functions. We study the linear case and the Kleinrock function and the BPR function in the nonlinear case. In Chapter 5, we propose a approximation scheme that uses compound travel time functions.

Chapter 4

Solution Methods

Contents

In the last decades, MCF have experienced impressive progresses in the solution methods. With a wide variety of applications and possibly huge size of the problems found in practice, MCF has been a motivating and challenging field for many researchers. Many approaches have been proposed in the literature to solve linear and/or nonlinear MCF. To be efficient, most solution methods exploit the special underlying network structure of the constraints. Direct methods adapt linear programming codes. Others decompose the problem using either linearization techniques or relaxation schemes. In this section, we shortly describe the underlying ideas of the main methods. We classify them in two categories: the primal and the dual methods. The first ones, that are described in the first section, solve the primal formulation of the problem directly, while the second ones work on the dual and appear in the second section. We refer the reader to [4, 102, 103] for other surveys on methods.

4.1 Primal solution approaches

In this section, we present shortly the main primal solution methods used in the literature. The most popular primal approaches are either direct or based on Dantzig-Wolfe decomposition in the linear case and, in the nonlinear case Frank-Wolfe or Simplicial decompositions are more appropriate.

4.1.1 Direct methods

Direct approaches consist in solving mainly the linear multicommodity flow problem with a linear programming code exploiting the special underlying block-network structure of the constraint matrix (2.5). Few methods have been extended to the nonlinear case. We shortly mention two main direct approaches.

The basis partitioning method, presented in [4], uses the spanning tree interpretation of any linear programing basis. Performing appropriate changes of variables, the programing basis for the MCF can be transformed in a special structure. This structure contains one basis for each commodity and a small basis for coupling constraints. Then the basis partitioning method is divided in two steps. It first uses general linear programing to compute the simplex multipliers corresponding to the coupling constraints. This computation only requires the inverse of a small basis. In the second step, a substitution procedure leads to find simplex multipliers for independent single commodity flow problems. Using special network simplex approach, this computation appears to be very efficient. In [29], a nonlinear primal partitioning, named PPRN, is adapted for nonlinear MCF. See [63, 74, 45] for other experiments with primal partitioning techniques.

We note that since the Simplex method needs to compute the inverse of the basis corresponding to the coupling constraints, the basis partitioning method is very efficient when the number of coupling constraints is small (see [4]). In the case of large number of coupling constraints, enhancements have been proposed by Castro and Nabona [29] and Mamer and McBride [93] using formulation (2.10) in which coupling constraints represent capacity constraints. The authors exploit the following observation. When the number of capacity constraints is large, the number of saturated arcs is mostly small at the optimum. As the unsaturated capacity constraints are not relevant in the solution, one may remove these constraints from the model. It results in solving a much smaller dimension problem. Unfortunately, that set of arcs is unknown. Strategies have then been developed to estimate the set of

unsaturated arcs during the process and to efficiently change the dimension of the working inverse from one pivot to the next. In [94], McBride observes a reduction from 17% to 70% of the size of the working inverse on several instances. In Chapter 5, we develop a similar strategy, called active set strategy, in a cutting plane framework.

Interior point methodology is an alternative direct approach to the simplex method. Specialized interior point methods also exploit the special structure of the problem to solve efficiently the linear system of equations at each iteration of the process. Most solution methods [6, 25, 107] use the preconditioned conjugate gradient solver, (which is originally appropriate for solving the single commodity flow problem). Castro [25] has implemented this technique making his interior point method competitive. Nevertheless, in [26], Castro shows that Simplex based solvers outperform specialized interior-point methods for medium and large scale instances. In that paper, the author solves problems in which commodities have multiple origin nodes and destination nodes. In [27], Catro extends this approach for solving nonlinear MCF with convex and separable quadratic objective functions. Finally in [111], Schultz and Meyer use a barrier function in a interior point framework to decompose the problem.

4.1.2 Dantzig-Wolfe Decomposition

Dantzig-Wolfe decomposition (DW) [39] is a price directive scheme used to solve linear programs whose constraints are partitioned in two blocks. DW decomposition is also described as a column generation scheme. It replaces by variables the subset of constraints containing the majority of constraints. To be advantageous this set of constraints must have a particular structure that makes the linear program subject to these constraints easy to solve. Consider the linear MCF formulation (2.10),

$$\min_x \left\{ r^T x \mid Mx \leq c,\ x \in \mathcal{X} \right\}. \tag{4.1}$$

Let $\{X_i\}_{i \in I}$ be all the extreme points of the convex set $\mathcal{X}$. It is well-known that any point of $\mathcal{X}$ can be obtained by forming a convex combination of extreme points and that the number of extreme points is finite. Then, assuming that the set of extreme points is explicitly given, problem (4.1) is equivalent to the *Dantzig-Wolfe master program*

$$\min_\lambda \left\{ r^T \sum_{i \in I} \lambda_i X_i \mid M(\sum_{i \in I} \lambda_i X_i) \leq c,\ \sum_{i \in I} \lambda_i = 1,\ \lambda > 0 \right\}. \tag{4.2}$$

Since we assign a variable to each extreme point and since the number of extreme points may be huge, problem (4.2) has too many variables to be solved directly. The key idea of Dantzig-Wolfe algorithm is to generate extreme points, or columns, as needed and to apply a revised Simplex algorithm to the *restricted master program*

$$\min_{\lambda} \left\{ r^T \sum_{i \in I_k} \lambda_i X_i \mid M(\sum_{i \in I_k} \lambda_i X_i) \leq c,\ \sum_{i \in I_k} \lambda_i = 1,\ \lambda > 0 \right\}. \tag{4.3}$$

The set $\{X_i\}_{i \in I_k}$ is assumed to be a small subset of the extreme points of $\mathcal{X}$ generated at the kth iterate. Assuming that (4.3) is feasible, we denote u_k and z_k the optimal dual variables in (4.3) associated to the constraints $M(\sum_{i \in I_k} \lambda_i X_i) \leq c$ and $\sum_{i \in I_k} \lambda_i = 1$, respectively. If the reduced costs σ_i of variables λ_i defined by

$$\sigma_i = r^T X_i + u_k^T M X_i - z_k, \quad \forall i \in I,$$

are all positive, the optimal solution of the restricted problem (4.3) is also the optimal solution for the full master (4.2). Otherwise the columns with negative reduced costs are added in the restricted master problem. It is easy to show that finding the column with the minimum reduced cost consists of solving the linear *Dantzig-Wolfe subproblem*

$$\min_{x} \left\{ (r + M^T u_k)^T x - z_k \mid x \in \mathcal{X} \right\}. \tag{4.4}$$

Obviously, Dantzig-Wolfe decomposition is appropriate only if subproblem (4.4) is easy to solve. In the linear MCF case, the matrix of constraints X is block angular. Solving (4.4) leads either to a sum of transshipment problems or to a sum of shortest path problems in the simplest case.

The Dantzig-Wolfe decomposition algorithm can be summarized as follows.

Algorithm 1: Dantzig-Wolfe decomposition

1. Find an initial set of extreme points, or columns, $\{X_i\}_{i \in I_0}$.
2. Solve the restricted master problem (4.3) and obtain the optimal dual variables u_k and z_k.
3. Solve the Dantzig-Wolfe subproblem (4.4). Pick a column X_k with minimum reduced cost.
   - If the minimum reduced cost $\sigma_k \geq 0$, then stop.
   - Else update the subset $\{X_i\}_{i \in I_{k+1}} = \{X_i\}_{i \in I_k} \cup X_k$ and go to step 2.

Note that it is necessary to initialize the set of extreme points I_0 that makes the restricted master problem (4.3) feasible. When I_0 is not given a priori, a Phase I procedure is required. We call I_0 the feasible initial set. Considering the general formulation problem

$$\min \left\{ c^T x \mid Ax = b,\ x \geq 0 \right\},$$

the standard Phase I procedure introduces artificial variables s in the problem and solves the linear program

$$\min \left\{ e^T (s^+ + s^-) \mid Ax + s^+ - s^- = b,\ x \geq 0,\ s^+ \geq 0,\ s^- \geq 0 \right\}, \tag{4.5}$$

where e is an one vector. Assuming that the original problem is feasible, then a Simplex algorithm applied to Phase I problem (4.5) will give a optimal value of zero and then a feasible solution for the original problem.

Let us now show how we can take advantage of the block angular structure of constraints in MCF (see Remark 1 page 16). The special network structure provides a natural partition of the constraints. The subproblem breaks down into several subproblems of much smaller dimension. Let us introduce the disaggregate formulation of (4.1)

$$\min_x \quad \sum_{\kappa \in \mathcal{K}} (r^\kappa)^T x^\kappa \tag{4.6a}$$

$$\sum_{\kappa \in \mathcal{K}} M^\kappa x^\kappa \leq c, \tag{4.6b}$$

$$x^\kappa \in X^\kappa, \quad \forall \kappa \in \mathcal{K}. \tag{4.6c}$$

Problems (4.1) and (4.6) are similar. Let $\{X_i^\kappa\}_{i\in I^\kappa}$ be the set of all the extreme points of the convex set $\mathcal{X}^\kappa$. In view of the above definition, one may write the restricted master program such as

$$\min_{\lambda>0} \quad \sum_{\kappa\in\mathcal{K}}(r^\kappa)^T \sum_{i\in I_k^\kappa} \lambda_i^\kappa X_i^\kappa \tag{4.7a}$$

$$\sum_{\kappa\in\mathcal{K}} M^\kappa(\sum_{i\in I_k^\kappa} \lambda_i^\kappa X_i^\kappa) \leq c, \tag{4.7b}$$

$$\sum_{i\in I_k^\kappa} \lambda_i^\kappa = 1, \quad \forall\kappa \in \mathcal{K}. \tag{4.7c}$$

This master problem has the same number of coupling constraints as the master (4.3), but now the subproblem (4.4) is separable in n_c independent subproblems, i.e., one per commodity. Letting u_k be the optimal dual variables associated to the coupling constraints (4.7b) and z_k^κ denote the optimal dual variables of (4.7c) associated to the commodity κ, the subproblem for the commodity κ is

$$\min_{x^\kappa} \left\{ (r^\kappa + (M^\kappa)^T u_k)^T x^\kappa - z_k^\kappa \mid x^\kappa \in \mathcal{X}^\kappa \right\}. \tag{4.8}$$

Depending on the definition of $\mathcal{X}^\kappa$, solving (4.8) results either in a simple shortest path problem or in a transshipment problem.

Dantzig-Wolfe decomposition permits to solve large scale linear problems. When phase I is completed the scheme has the important advantage to maintain a feasible solution for the problem during the process. In [4], the authors point out that Dantzig-Wolfe approach is efficient in terms of the number of iterations but not necessarily in CPU time. The reason is that solving the linear master restricted problem at each iteration turns out to be too time consuming to make the method competitive on large instances.

Many works in the literature use the Dantzig-Wolfe decomposition to solve linear MCF. Farvolden et al. [45] applied it to an arc-chain formulation of the MCF. In [30], Chardaire and Lisser compared Dantzig-Wolfe decomposition with direct approaches and also with ACCPM [58] on oriented MCF. They concluded that using a fast simplex-based linear programming code to solve the restricted master program is the fastest alternative on small and medium size problems.

4.1.3 Frank-Wolfe method

The Frank-Wolfe algorithm [50] is a linearization method for nonlinear optimization problems with a convex objective function and linear constraints. It consists of finding the optimum by successive linearizations of the objective function that yield useful descent directions. To take full advantage of the procedure the linearized problem must be easy to solve. In the MCF context, the Frank-Wolfe method is applied to the nonlinear program (2.12), i.e.,

$$\min \{g(y) \mid y \in \mathcal{Y}\}. \tag{4.9}$$

Let y^0 be a feasible solution for (4.9). At the iteration k, the method linearizes the objective function in y^k and solves the *direction finding subproblem*

$$Y^k = \arg\min \left\{\nabla g(y^k)^T y \mid y \in \mathcal{Y}\right\}, \tag{4.10}$$

to obtain a feasible direction $(Y^k - y^k)$ at y^k. Problem (4.10) is a simple linear problem. It can be solved with a Simplex algorithm. If $\nabla g(y^k)^T(Y^k - y^k) \geq 0$, then y^k satisfies the optimal stopping criterion. Otherwise, the algorithm computes the optimal step along the descent direction by solving the *line search problem*

$$\lambda^k = \arg\min \left\{g(y^k + \lambda(Y^k - y^k)) \mid 0 \leq \lambda \leq 1\right\}. \tag{4.11}$$

The Frank-Wolfe algorithm can be stated as follows:

Algorithm 2: Frank-Wolfe algorithm

1. Find a feasible flow vector y^0.
2. Solve the direction finding subproblem (4.10) and obtain a feasible direction $(Y^k - y^k)$.
3. If $\nabla g(y^k)^T(Y^k - y^k) \geq 0$, then stop.
4. Perform the line search problem (4.11), update $y^{k+1} = y^k + \lambda^k(Y^k - y^k)$ and go to step 2.

The Frank-Wolfe algorithm is easy to understand and implement. Another positive aspect of the method is the small memory storage requirement. Though Frank-Wolfe makes big progresses in the first iterations, the method is known

to have a slow convergence rate [103, 15]. When approaching optimality the method tends to zigzag. Several enhancements have been proposed to accelerate the convergence. In [123], Wolfe proposed to generate an additional feasible direction in step 1. The PARTAN method, introduced by Shah et al. [113], also has been proposed to avoid the zigzag behavior.

Frank-Wolfe procedure is attractive for nonlinear MCF without mutual capacity constraints because the direction finding subproblem becomes easy. There, it turns out to be an unconstrained linear MCF that is separable in independent shortest path problems (or transshipment flow problems). When applied to a problem with mutual capacity constraints, the subproblem becomes a linear MCF with mutual capacities which is too expensive to be solved repeatedly. In the case of objective function with a vertical asymptote, the capacities are implicitly defined. The standard technique to cope with functions with a vertical asymptote consists in approximating the objective function by a function with an unbounded domain. Some variant of the Frank-Wolfe algorithm has been applied to nonlinear MCF, e.g. [47, 51, 86]. The convergence of the method has been improved by Weintraub et al. [122]. More recently, Daneva and Lindberg [38] reported dramatic acceleration using a conjugate gradient scheme. Fratta et al. [51] introduced the *Flow Deviation Method*, which is similar to the Frank-Wolfe algorithm to solve communication network problems.

4.1.4 Simplicial decomposition

The simplicial decomposition [69, 120] can be viewed as a combination of Dantzig-Wolfe decomposition [39] and Frank-Wolfe algorithm [50] to solve nonlinear programs. In the simplicial decomposition, extreme points, or columns, are generated by solving a linear subproblem similar to the direction finding subproblem of Frank-Wolfe. As in the Dantzig-Wolfe approach, a master program, is defined by a restricted set of extreme points and is solved to generate a new point.

This approach is well fitted to solve the nonlinear MCF formulation (2.12), i.e.,

$$\min \{g(y) \mid y \in \mathcal{Y}\}, \tag{4.12}$$

where $\mathcal{Y} \subset \mathbb{R}^m$. Suppose that in the kth iteration, a set of extreme points (or columns) $\{Y^i\}_{i \in I_k}$ has been generated by the Frank-Wolfe subproblem (4.10). Then the method proposes to find an optimal linear combination of

the columns by solving the master problem

$$\min_{\lambda} \left\{ g(\sum_{i \in I_k} \lambda_i Y_i) \mid \sum_{i \in I_k} \lambda_i = 1,\ \lambda \geq 0 \right\}. \tag{4.13}$$

This step can be viewed as a generalization of the line search problem in the Frank-Wolfe algorithm. The resulting linear combination corresponds to the new point. At that point, we then perform a standard Frank-Wolfe step (4.10) to define a new extreme point.

The simplicial decomposition can be summarized as follows:

Algorithm 3: Simplicial decomposition

1. Find a feasible flow vector y^0.
2. Solve the Frank-Wolfe subproblem (4.10) and obtain a extreme point Y^k.
3. If $\nabla g(y^k)^T(Y^k - y^k) \geq 0$, then stop.
4. Update the subset $\{Y_i\}_{i \in I_k} = \{Y_i\}_{i \in I_{k-1}} \cup Y_k$. Solve the master problem (4.13), update $y^{k+1} = \sum_{i \in I_k} \lambda_i Y_i$ and go to step 2.

According to Carathéodory's theorem, at most $m+1$ columns are necessary to represent an optimal solution of the full master problem (4.13). In practice, this number is often too large to be useful. A stronger bound for the number of needed extreme points is given by Hearn et al. [65] by defining a restricted simplicial decomposition. Numerical experiments in [65, 66] demonstrated efficiency of this restricted simplicial decomposition to solve traffic assignment problems.

Bertsekas [20] proposed a scaled projected Newton method to find an improved allocation in the master problem. This approach, named Projection method, has been applied to MCF in [21, 22]. The method developed by Bar-Gera [15] is an origin-based algorithm conceptually similar to [21]. In the comparative study carried in [102], the projected Newton method appeared to be one of the most efficient method on small and medium size problems. An other projection method has been proposed by M. Schwartz et al. [112] to solve message routing problems. Finally, Larsson et al. exploited the special structure of the traffic assignment problem using a disaggregate version of the simplicial decomposition [82].

Let us just mention that, as in the case of Frank-Wolfe algorithm, the simplicial decomposition cannot handle arc capacity constraint directly. One must replace constraint violation by a fixed penalty function.

4.1.5 Other methods

Another approach is based on *resource-directive decompositions* [4, 76, 77, 75, 54]. It consists of finding an optimal allocation of the mutual capacities for each commodity. The initialization step of the iterative method assigns the mutual capacity for each commodity and solves the associated independent single commodity flow problems. Then at each iteration, the resource-directive approach uses sensitive information about each subproblem to reallocate the mutual capacities. Subgradient method has been used in several studies [10, 67, 77, 115] to find the optimal capacity allocation. In a comparative study, Assad [10] reports that resource-directive algorithms converge quickly for small problem instances but are outperformed by price-directive method for larger problems.

Combinatorial approximation algorithms have also been developed to solve MCF. Some of the approximation algorithms may guaranty theoretical results on computational complexity [46, 72, 87] and appear to be competitive with other methods [61, 110].

4.2 Dual solution approaches

In this section, we present shortly the main dual solution methods found in the literature. Lagrangian relaxation is the most common dual method used to solve linear and nonlinear MCF. Augmented Lagrangean algorithms and the proximal decomposition method have also been applied to MCF.

4.2.1 Lagrangian relaxation

Lagrangian relaxation may be used to solve linear or nonlinear multicommodity flow problems. It consists of bringing the coupling constraints associated to Lagrangian multipliers into the objective. Then the approach finds appropriate prices, or Lagrangian multipliers, that achieve the dual optimum. The method may be directly applied on the general formulation of the problem (2.7), i.e.,

$$\min_{x,y} \{g(y) + \langle r, x\rangle \mid Mx = y \text{ and } x \in \mathcal{X}\}. \tag{4.14}$$

Let u be the dual multipliers associated to the set of constraints $Mx = y$, Lagrangian relaxation yields a Lagrangian dual problem of much smaller dimension

$$\max_{u} L(u), \tag{4.15}$$

where

$$L(u) = \min_{x,y} \{g(y) + \langle r, x\rangle + \langle u, Mx - y\rangle \mid x \in \mathcal{X}\}. \tag{4.16}$$

Problems (4.15) and (4.16) are called the master problem and the subproblem, respectively. Solving the master problem (4.15) gives a lower bound for the primal problem (4.14). By duality theory, the lower bound of the optimal dual solution is equal to the optimal value of (4.14). Then Lagrangian relaxation appears to be an useful approach to find the optimal objective value for MCF problems but does not directly provide an optimal primal solution point.

It is well-known that the Lagrangian dual (4.15) is nondifferentiable. When the objective function g is linear, the subproblem is a sum of simple transshipment flow problems (or shortest path problems) and the master problem may be described by a concave piecewise linear function.

For the sake of simplicity, let us consider the linear case (2.10) where the dual subproblem is defined by

$$L(u) = \min_{x} \{\langle r, x\rangle + \langle u, Mx - c\rangle \mid x \in \mathcal{X}\}. \tag{4.17}$$

Note that problem (4.17) will only generate extreme points of $\mathcal{X}$. Assuming that $\{X_i\}_{i\in I}$ represents the set of all the extreme points of the convex set $\mathcal{X}$, the maximization of L is then equivalent to the full master problem

$$\max_{u,z} \quad z \tag{4.18a}$$

$$z \leq \langle r, X_i\rangle + \langle u, MX_i - c\rangle, \quad i \in I, \tag{4.18b}$$

where z is the epigraph variable associated to the epigraph of L. Theorem 3 shows that Lagrangian relaxation scheme is very similar to Dantzig-Wolfe decomposition.

Theorem 3. *The full master problem (4.18) of the Lagrangian relaxation is equivalent to the dual of the Dantzig-Wolfe master program (4.2).*

Proof. The proof is immediate just writing the dual of (4.18) or (4.2). □

We also notice that the Lagrangian subproblems correspond to the Dantzig-Wolfe subproblems. In fact, we can view Dantzig-Wolfe decomposition as a method for solving the Lagrangian multipliers. Since the full master problem is impractical, methods solving the dual problem generate extreme points as needed.

Most standard methods used for solving the Lagrangian dual problem can be classified as cutting plane methods (CPM). Kelley's cutting plane method [73] is appealing because the master problem is linear, but it often converges very slowly. Notice that this approach corresponds to the dual of the Dantzig-Wolfe decomposition scheme [39]. The bundle method [88] has been used in the context of linear MCF [49] to remedy this convergence drawback. Good numerical results have been obtained with the analytic center cutting plane method (ACCPM) [60]. In that paper, ACCPM was used in a disaggregate mode, that is with as many objective components as the number of commodities. Finally, we mention a subgradient method, presented in [4], to solve (4.15). An important improvement of the method [99] has been proposed by Nesterov. As we know, the method has not yet been applied on MCF but it seems to be a promising approach. We review these main cutting plane methods in Chapter 5.

When g is a nonlinear function of the total flow, it is still possible to describe the master problem with polyhedral approximations. We observe that the subproblem (4.16) is separable in variables x and y such as

$$L(u) = \min_x \{\langle r, x\rangle + \langle u, Mx\rangle \mid x \in \mathcal{X}\} + \min_y \{g(y) - \langle u, y\rangle\}.$$

The first component may be described by polyhedral approximation while the second one is smooth and can be computed in closed form. Then the dual problem (4.15) may also be solved using cutting plane methods. In Chapter 5, we propose a different approach. We further propose to exploit the second order information of the smooth function in a cutting plane framework, i.e., ACCPM. This approach will be detailed in Chapters 5 and 6.

4.2.2 Augmented Lagrangian relaxation

The augmented Lagrangian relaxation combines two schemes, the standard Lagrangian relaxation and an exterior penalty method. A nonlinear penalty is introduced in order to improve the dual convergence and obtain a primal optimal solution. Let us consider the general formulation of the MCF (2.7)

with a monotone increasing function g,

$$\min_{x,y} \{g(y) + \langle r, x\rangle \mid h(x,y) \leq 0 \text{ and } x \in \mathcal{X}\}, \tag{4.19}$$

where $h(x,y) = Mx - y$, is a vector defined in $\mathbb{R}^{n_a}$. Let $u \geq 0$ be the vector of Lagrangian multipliers and $v > 0$ be the penalty parameter, the augmented Lagrangian dual problem is defined by

$$\max_{u \geq 0} L_v(u). \tag{4.20}$$

The augmented Lagrangian subproblem is

$$L_v(u) = \min_{x \in \mathcal{X}, y} L_v(u, x, y), \tag{4.21}$$

where

$$L_v(u,x,y) = g(y) + \langle r, x\rangle + \sum_{a \in \mathcal{A}} u_a h_a^+(x_a, y_a, u_a, v) + \sum_{a \in \mathcal{A}} \frac{v}{2} h_a^+(x_a, y_a, u_a, v)^2,$$

and

$$h_a^+(x_a, y_a, u_a, v) = \max\{h_a(x_a, y_a), -\frac{u_a}{v}\}.$$

A solution of the subproblem is denoted

$$x(u,v) = \arg\min_{x \in \mathcal{X}, y} L_v(u,x,y).$$

To achieve convergence to the dual optimal solution, the Lagrangian multipliers and the penalty parameter are updated iteratively. In the kth iteration, the updating formulas are

$$u_a^{k+1} = \max\{0, u_a^k + v h_a(x_a(u_a, v), y_a)\}, \tag{4.22}$$

and

$$v^{k+1} = \begin{cases} \min(\beta_v v^k, v_{max}), & \text{if } ||h_a^+(x_a^k, y_a^k, u_a^k, v^k)|| > \\ & \gamma ||h_a^+(x_a^{k+1}, y_a^{k+1}, u_a^{k+1}, v^{k+1})||, \\ v^k, & \text{otherwise,} \end{cases} \tag{4.23}$$

where $\beta_v > 1$ and γ are constant factors. In [81], Kort and Bertsekas proved the convergence of the method when $v^0 > 0$ and $v^{k+1} \geq v^k$.

The augmented Lagrangian algorithm can be stated as follows.

Algorithm 4: Augmented Lagrangian relaxation

1. Initialize the dual variable u^0 and the parameters v^0, v_{max}, γ and β^0.
2. Solve the subproblem (4.21) and if termination criteria is satisfied then stop.
3. Update variables and parameters with (4.22) and (4.23) and go to step 2.

Note that, in step 2, the subproblem (4.21) is a nonlinear MCF. The standard way to solve this problem is to apply a disaggregate simplicial decomposition [84, 83]. In this approach, the difficulty of the subproblem to be solved at each iteration is balanced by a better convergence of the master problem than for the standard Lagrangian relaxation. In a recent contribution, Larsson and Yuang [84] apply an augmented Lagrangian algorithm to solve linear MCF. They are able to find solutions with reasonable precision on very large problem instances. They also show that their method outperforms the Dantzig-Wolfe decomposition approach and the combination of Lagrangian relaxation and a bundle implementation. An augmented Lagrangian technique has been used in [83] to solve nonlinear traffic assignment problems with link capacity constraints.

4.2.3 Proximal Decomposition method

We conclude this review by mentioning the Proximal Decomposition method of [92] which is a specialization of the Partial Inverse method of Singarn [118] for separable convex problems. The main idea of the method is to represent the coupling constraints by a product of subspaces corresponding to the copy of primal-dual variables. The algorithm performs two steps iteratively. First, it regularizes the objective function by adding a quadratic term which depends on primal-dual solution of the previous iterate. Then, the method projects the solution on the corresponding subspaces. The Proximal Decomposition method has been applied on telecommunication routing problems [92, 102]. The method appears to be competitive with other methods but it is slow when the number of commodities is large.

4.3 Conclusion

We notice that it is not evident to know which solution method (direct approach or decomposition scheme) is better suited for a particular type of problem. The choice might depend on the size of the problem, the number of commodities, the number of coupling constraints, the sparsity of the network matrix or the number of used or saturated arcs in the optimal solution. In [26], Castro also observes that there does not exist a decision rule to know in advance the best method for each particular problems. He comments that it is still an important work to be done in MCF.

In our work, we apply the Lagrangian relaxation scheme which has the advantage of solving both linear and nonlinear cases of MCF problems. The efficiency of this approach mainly depends on the cutting plane method used to solve the dual problem, hence an important focus of the thesis.

Part II

Constrained ACCPM and active set strategy

Chapter 5

Lagrangian relaxation and active set strategy

Contents

The literature proposes many different approaches to solve the multicommodity flow problem (see Chapter 4). One of the most popular is based on price-directive decomposition, i.e., a Lagrangian relaxation approach which turns out to be equivalent to a column generation scheme. In the case of MCF, column generation amounts to working on a sequence of restricted versions of the path-flow formulation of the problem. At each iteration, the method generates a solution for each commodity, with respect to arc lengths equal to the current marginal value of the delay or congestion function. The method then strives to allocate the flows on the generated paths in an optimal way.

In this chapter, we introduce the Lagrangian relaxation of the general multicommodity flow problem introduced in Chapter 2. We point out that the objective of the Lagrangian dual is a sum of a nonsmooth function and a smooth one. The first component may be described by polyhedral approximation, while the second one can be computed in closed form. We further propose to exploit the second order information of the smooth function to enhance efficiency of the solution method.
Then, we focus on the *compound congestion function* which has the following characteristic: it is linear near the origin. We show that there exists an optimal partition of the Lagrangian dual variables for this kind of function and that an active set strategy may be used to guess it. This optimal partition leads to a reduced dual problem formulation. Finally, we introduce an approximation scheme which replaces the nonlinear congestion function by a compound one. We will show that this feature permits to display an active set strategy.

5.1 Lagrangian Relaxation

Let us consider the general problem

$$\min \quad g(y) + \langle r, x \rangle \tag{5.1a}$$
$$Mx = y, \tag{5.1b}$$
$$x \in \mathcal{X}. \tag{5.1c}$$

We assume that $g : \mathbb{R}^m \to \mathbb{R}$ is convex, M is a $m \times n$ matrix, r is a n vector, while $\mathcal{X} \subset \mathbb{R}^n$ is convex.

Remark 6. *We easily identify problem (5.1) with the general formulation of MCF (2.7) introduced in Chapter 2. In multicommodity flow problems on oriented graphs, $\mathcal{X}$ is defined as a set of network flow constraints (one per commodity). The matrix M collects the flows on the individual arcs; it is thus made of zeroes and ones.*

If the dimension of x is large, as it is the case in MCF problems, it becomes difficult to apply a direct method, even when the problem falls into the realm of structural programming (see [98]), e.g., g is self-concordant and $\mathcal{X}$ is endowed with a self-concordant barrier.

Relaxing (5.1b) yields a concave programming problem in the dual variables $u \in \mathbb{R}^m$ associated with the constraints $y = Mx$. The Lagrangian dual problem is

$$\max_u f(u), \tag{5.2}$$

where f is defined by

$$f(u) = \min_{y,x}\{g(y) + \langle r, x\rangle + \langle u, Mx - y\rangle \mid x \in \mathcal{X}\}. \tag{5.3}$$

By duality, this Lagrangian dual problem has the same optimal value as (5.1). Since $m \ll n$, the transformed problem has much smaller dimension: it is potentially solvable by a suitable convex optimization method. We review the main ones in Section 5.3.

A quick inspection shows that (5.3) is separable in the variables y and x. Then, (5.3) can be written as

$$f(u) = f_1(u) + f_2(u), \tag{5.4}$$

where

$$f_1(u) = \min_{x}\{\langle M^T u + r, x\rangle \mid x \in \mathcal{X}\},$$

and

$$f_2(u) = \min_{y}\{g(y) - \langle u, y\rangle\}.$$

In view of our assumption on $\mathcal{X}$, $f_1(u)$ can be routinely computed for arbitrary values of u. Since f_1 is defined as the point-wise minimum of a collection of linear functions, f_1 is concave but usually nondifferentiable. Besides, if

$$x(u) = \operatorname{argmin}\{\langle M^T u + r, x\rangle \mid x \in \mathcal{X}\},$$

then

$$f_1(u') \geq \langle Mx(u), u'\rangle + \langle x(u), r\rangle = f_1(u) + \langle Mx(u), u' - u\rangle, \quad \forall u' \in \mathbb{R}^m. \tag{5.5}$$

This shows that $Mx(u)$ is an anti-subgradient of $f_1(u)$ at u, that is $Mx(u) \in -\partial(\text{-}f_1(u))$.

Akin, the function $f_2(u)$ is the point-wise minimum of a collection of affine functions of u. It is thus concave and one may construct an inequality similar to (5.5). Actually, we can get more. From the definition, we observe that $f_2(u)$ is the opposite of the Fenchel conjugate $g_*(u)$ of g. In the cases under study, $g_*(u)$ can be given in closed form and it also appears to be twice continuously differentiable. We certainly want to exploit this property when it is verified, and devise more efficient algorithms to solve the Lagrangian dual problem.

Let us put forth two conditions of MCF that are of considerable help in solving (5.2).

Condition 1. *The linear programming problem*

$$\min\{\langle c, x\rangle \mid x \in \mathcal{X}\},$$

can be solved at low computational cost.

In other words, $f_1(u)$ can be computed routinely, without excessive burden on the overall algorithm. In MCF, f_1 is either a sum of shortest path problems or transshipment problems.

Condition 2. *The congestion function $g(y)$ is separable, i.e., $g(y) = \sum_{i=1}^{m} g_i(y_i)$. The functions g_i are nonnegative, convex, monotonically increasing and $\operatorname{dom} g_i \subset \mathbb{R}_+$. Moreover, the convex conjugate $(g_i)_*$ can be computed in closed form.*

Let us explore an immediate consequence of Condition 2. The first order optimality conditions for problem (5.1) are

$$0 \in \partial g(y) - u, \tag{5.6}$$

$$M^T u \in -\mathcal{N}_{\mathcal{X}(x)}, \tag{5.7}$$

where $\mathcal{N}_{\mathcal{X}(x)}$ is the normal cone of $\mathcal{X}$ at x. The right derivative $g'_+(y)$ of g at $y = 0$ is well-defined. Since g is monotone increasing, condition (5.6) implies that the constraint

$$u \geq g'_+(0), \tag{5.8}$$

is always met at the optimum. It is nevertheless convenient to introduce this redundant constraint in the formulation of problem (5.2).

To summarize, the Lagrangian dual problem we propose to solve is a maximization problem subject to lower bounds on variables:

$$\max\{f(u) = f_1(u) + f_2(u) \mid u \geq g'_+(0)\}. \tag{5.9}$$

The standard way to solve (5.9) is to apply a cutting plane method that describes iteratively the objective function f with polyhedral approximations. We give some examples of such methods in Section 5.3. In Chapter 6, we enhance a cutting plane scheme to exploit the available second order information.

Remark 7. *In the constrained system optimum problem, introduced in Chapter 3 page 38, Condition 1 is not satisfied. The subproblem is a NP-hard problem, i.e., a sum of constrained shortest path problems. In that situation, the optimization process may not be carried to its end and the computed objective value may be a strict lower bound for the true value $f_1(u)$. However, it is generally the case that it produces a polyhedral approximation form that underestimates the function f_1. In Chapter 6, we propose advanced topics in ACCPM that deals with this undesirable situation.*

5.2 Extension to elastic demand model

In this section, we extend the analysis of the above section for the traffic assignment problem with elastic demand introduced in Chapter 3. Let us consider the problem

$$\begin{aligned} \min \quad & g(y) + h(\delta) && \text{(5.10a)} \\ & Mx = y, && \text{(5.10b)} \\ & x \in \mathcal{X}(\delta). && \text{(5.10c)} \end{aligned}$$

Relaxing constraints (5.10b), the dual objective boils down to

$$f(u) = f_1(u) + f_2(u), \tag{5.11}$$

where

$$f_1(u) = \min_{x,\delta}\{h(\delta) + \langle M^T u, x\rangle \mid x \in \mathcal{X}(\delta)\},$$

and

$$f_2(u) = \min_{y}\{g(y) - \langle u, y\rangle\}.$$

As in the previous section, the function f_2 can be given in a closed form. Let us now show that f_1 may again be easily computed and described by polyhedral approximations. This problem is equivalent to a two-stage minimization

$$f_1(u) = \min_{\delta}\left\{h(\delta) + \min_{x}\{\langle M^T u, x\rangle \mid x \in \mathcal{X}(\delta)\}\right\}.$$

In our context, the optimal solution of the inner problem of f_1 is a shortest path problem with a flow demand δ. Indeed, the path is independent of δ; the flow only is affected by δ. Let $\xi(u)$ be the boolean vector associated with that path. The optimal solution of the inner problem in f_1 takes the form $\langle M^T u, \xi(u)\delta\rangle$. The minimization in δ can then be performed, usually in explicit form when h is given a functional form.

To show that f_1 satisfies a subgradient inequality, let us start with the observation that for any $u' \neq u$, the shortest path property says

$$v' = \langle M^T u', \xi(u')\rangle \leq \hat{v} = \langle M^T u', \xi(u)\rangle,$$

and

$$\delta(v) = \arg\min(h(\delta) + v\delta).$$

Since $\delta > 0$ and $v' \leq \hat{v}$,

$$\begin{aligned} \min_{\delta}\{h(\delta) + v'\delta\} &\leq \min_{\delta}\{h(\delta) + \hat{v}\delta\}, \\ &\leq h(\delta(v)) + \hat{v}\delta(v), \\ &= h(\delta(v)) + v\delta(v) + (\hat{v} - v)\delta(v), \\ &= \min_{\delta}(h(\delta) + v\delta) + (\hat{v} - v)\delta(v). \end{aligned}$$

Replacing v' and v by their values in the above inequality, one gets

$$\begin{aligned} \min_{\delta}\{h(\delta) + \langle M^T u', \xi(u')\rangle\} \leq \min_{\delta}(h(\delta) + \langle M^T u, \xi(u)\rangle) \\ +(u' - u)^T M\xi(u)\delta(\langle M^T u', \xi(u)\rangle). \end{aligned}$$

It follows that $M\xi(u)\delta(\langle M^T u', \xi(u)\rangle) \in \partial f_1(u)$ is a subgradient of f_1 at u. In the considered applications, the function h is convex and differentiable. Therefore, the function $\min_\delta(h(\delta) + v\delta)$ achieves its minimum at

$$\delta(v) = -(h')^{-1}(v).$$

To summarize, the approach is mostly the same as in the previous section. It differs only in the way of describing f_1 by polyhedral approximations. In that case, the inner minimization problem f_1 appears to be a two-stage optimization.

5.3 Solution methods for solving the Lagrangian dual problem

In this section, we describe some of standard methods for non-differentiable convex optimization. For the sake of simpler notation, we consider in the rest of the section the general minimization problem

$$\min\{f(u) \mid u \in U\}, \tag{5.12}$$

where f is convex and $U \subset \mathbb{R}^m$ is a convex set. We assume that for any point $u_i \in U$, we can construct a supporting hyperplane of the objective function, i.e., a cutting plane, such that

$$f(u) \geq f(u_i) + \langle \xi_i, u - u_i\rangle, \quad \forall u \in U,$$

where $\xi_i \in \partial f(u_i)$.

5.3.1 Kelley's method

The cutting plane method proposed by Kelley [73] consists of building iteratively a tangentially piecewise linear approximation of the objective function and solving the approximate problem at each iteration. Let $\{u_i\}_{i\leq k}$ be a set of k points where cutting planes have been generated. Assuming that the cutting planes are good approximations of the objective function, the quality of the linear piecewise approximation is improved when a new cutting plane is added in the model. Then, the method converges when the approximation is precise enough close to the optimal solution. The piecewise linear approximation of the objective function f is defined by

$$\hat{f}_k(u) = \max_{i\leq k}\{f(u_i) + \langle \xi_i, u - u_i\rangle\}, \tag{5.13}$$

Kelley's method is stated as follows:

Algorithm 5: Kelley's method

1. Choose $u_0 \in U$.
2. Define $\hat{f}_k(u) = \max_{i\leq k}\{f(u_i) + \langle \xi_i, u - u_i\rangle\}$.
3. Compute $u_{k+1} = \arg\min_{u\in U} \hat{f}_k(u)$ and go to step 2.

Although Kelley's method is globally convergent under few assumptions, it appears to be disastrous in some applications. In the worse case, $O(\frac{1}{\epsilon^m})$ iterations are required to converge with an ϵ relative precision [96].

5.3.2 Bundle method

The bundle method, proposed by Lemaréchal [88, 89], is a dramatic improvement of Kelley's method. The piecewise linear approximation (5.13) is augmented by a quadratic term such that

$$u_{k+1} = \arg\min_{u\in U}\{\hat{f}_k(u) + \frac{1}{2t_k}||u - u_k||^2\},$$

where t_k is a parameter to be updated at each iteration. The move from u_k to u_{k+1} is validated if a sufficient descent of f is observed. Otherwise the current point does not change. The bundle method is stated in Algorithm 6.

Algorithm 6: Bundle method

1. Choose $u_0 \in U$.
2. Define $\hat{f}_k(u) = \max_{i \leq k}\{f(u_i) + \langle \xi_i, u - u_i \rangle\}$.
3. Compute $\bar{u}_{k+1} = \arg\min_{u \in U}\{\hat{f}_k(u) + \frac{1}{2t_k}||u - u_k||^2\}$,
   - $u_{k+1} = \bar{u}_{k+1}$, if a sufficient descent of f is observed.
   - $u_{k+1} = u_k$, otherwise.

   and go to 2.

There exists different versions of the bundle method [78, 90] which differ in the way of updating the parameter t_k and evaluating a sufficient descent. The pseudo-polynomiality convergence of the method has been proved by Lemaréchal et al. in [90]. The scheme requires $O(\frac{1}{\epsilon^2})$ iterations to converge with an ϵ relative precision.

5.3.3 Subgradient method

The subgradient method is a simple method minimizing the nondifferentiable function f such that

$$u_{k+1} = u_k - \alpha_k \xi_k,$$

where $\alpha_k > 0$ is a step size. The different implementations of the subgradient method differ in the way the step size is defined. Algorithm 7 describes the subgradient method.

Algorithm 7: Subgradient method

1. Choose $u_0 \in U$ and a sequence of step size $\{\alpha_k\}_{k=0}^{\infty}$.
2. Compute $u_{k+1} = u_k - \alpha_k \xi_k$ and repeat 2.

When the step lengths $\{\alpha_k\}_{k=0}^{\infty}$ are defined to be a divergent series such that

$$\lim_{k \to \infty} \alpha_k = 0, \text{ and } \sum_{k=0}^{\infty} \alpha_k = \infty,$$

the subgradient method converges. The method required $O(\frac{1}{\epsilon^2})$ iterations to converge with an ϵ relative precision. Recently, Nesterov [99] has proposed a important improvement of the gradient scheme. He proved than the bound on the number of iterations of the new scheme is $O(\frac{1}{\epsilon})$.

5.3.4 The analytic center cutting plane method

The analytic center cutting plane method (ACCPM), introduced by Goffin et al. [58], combines cutting plane scheme and interior point methodology. It defines the so-called localization set as a convex polyhedron

$$\mathcal{L}_k = \{(u, z) \mid z \geq f(u_i) + \langle \xi_i, u - u_i \rangle,\ z \leq \bar{\theta}\},$$

where z is the epigraph variable of the function f and $\bar{\theta}$ is an upper bound for the problem. Let s be the slack variables associated to the cutting planes, ACCPM computes the analytic center as the point that maximizes the product of the slack variables.

Note that if the localization set is unbounded, the analytic center does not exist. Traditionally, box constraints on variables are added in the model to fix this problem.

In Chapter 6, we extensively detail a constrained version of the analytic center cutting plane method. This version is augmented with a proximal term and handles the available second order oracle of the Lagrangian dual problem.

5.4 Congestion functions and their conjugates

In this section, we present the analytic forms of the conjugate functions associated to the congestion or delay functions used in MCF. Condition 2 makes valid an explicit calculation of the conjugate function g_* of the congestion function g

$$g_*(u) = \max_y \{\langle u, y \rangle - g(y)\}.$$

Note that the conjugate function is also separable in variables y. We have

$$g_*(u) = \sum_{a \in \mathcal{A}} g_{a*}(u_a) = \sum_{a \in \mathcal{A}} \max_{y_a} \{u_a y_a - g_a(y_a)\}.$$

In the sequel we shall name -g_* the negative conjugate. The computation for the congestion functions introduced in a previous section is straightforward.

Since the objective $f_2(u)$ is the opposite of the conjugate function, we give, in the next subsections, the functional form of $-g_{a*}$, its domain and its first and second derivatives associated to our congestion functions of interest. For the sake of simpler notation, we write in the rest of the section g for g_a, y for y_a, u for u_a, etc.

5.4.1 Linear function

Let us recall the linear function:

$$g(y) = ty, \quad \text{with } y \in [0, c]. \tag{5.14}$$

The opposite of the conjugate function of (5.14) is defined by

$$f_2(u) = -g_*(u) = c(t - u), \quad \text{with } u \geq 0. \tag{5.15}$$

Its first and second derivatives are immediate. We plot the linear function (5.14) and its conjugate (5.15) on Figure 5.1.

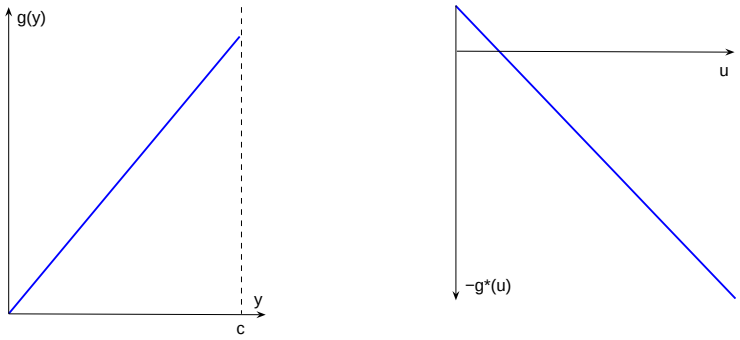

Figure 5.1: Linear function and its negative conjugate.

5.4.2 Kleinrock delay function

Let us write the Kleinrock delay function:

$$g(y) = \frac{y}{c - y}, \quad \text{with } y \in [0, c[. \tag{5.16}$$

The opposite of the conjugate function of (5.16) is defined by

$$f_2(u) = -g_*(u) = 1 + uc - 2\sqrt{uc}, \quad \text{with } u \geq \frac{1}{c}. \tag{5.17}$$

The Kleinrock function (5.16) and its conjugate (5.17) are plotted on Figure 5.2. Its first and second derivatives are given by

$$f_2'(u) = c - \sqrt{\frac{c}{u}}, \quad \text{and} \quad f_2''(u) = 0.5\sqrt{c}u^{-1.5}.$$

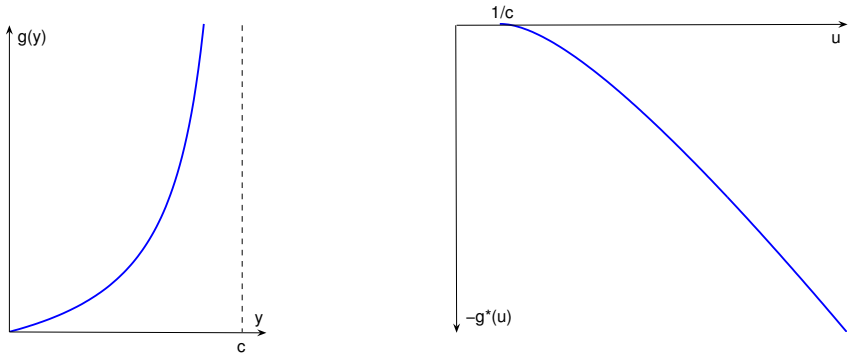

Figure 5.2: Kleinrock delay function and its negative conjugate.

5.4.3 BPR congestion function

Let us first recall the BPR congestion function:

$$g(y) = ry\left(1 + \frac{\alpha}{\beta+1}(\frac{y}{c})^{\beta}\right), \quad \text{with } y \geq 0. \tag{5.18}$$

The opposite of the conjugate function of (5.18) is defined by

$$f_2(u) = -g_*(u) = \frac{c(u-r)^{\frac{\beta+1}{\beta}}}{(\alpha r)^{\frac{1}{\beta}}}\frac{\beta}{\beta+1}, \quad \text{with } u \geq 0. \tag{5.19}$$

The BPR congestion function (5.18) and its conjugate (5.19) are plotted on Figure 5.3. Its first and second derivatives are given by

$$f_2'(u) = \frac{c}{(\alpha r)^{\frac{1}{\beta}}}(u-r)^{\frac{1}{\beta}}, \quad \text{and} \quad f_2''(u) = \frac{c}{\beta(\alpha r)^{\frac{1}{\beta}}}(u-r)^{\frac{1-\beta}{\beta}}.$$

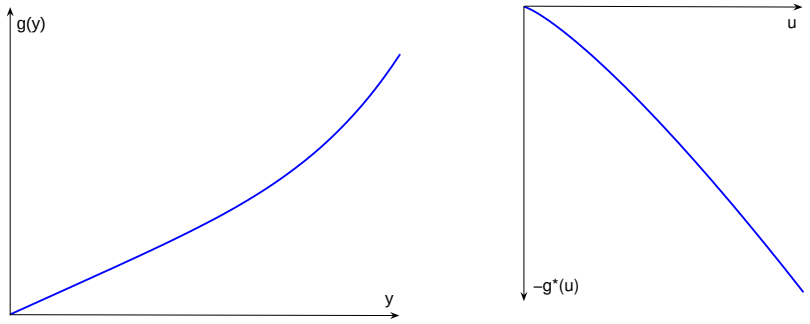

Figure 5.3: BPR congestion function and its negative conjugate.

5.4.4 Compound congestion functions

In this work, we shall also consider a more general class of congestion functions defined by

$$g(y) = \max_{y \in \operatorname{dom} \tilde{g}} \{ty, \tilde{g}(y)\}, \tag{5.20}$$

where $t \geq 0$ and $\tilde{g}(y)$ satisfies Condition 2 whose domain is a closed or half-closed interval of $\mathbb{R}_+$:

$$\operatorname{dom} \tilde{g} = [0, \bar{y}[\quad \text{or} \quad \operatorname{dom} \tilde{g} = [0, \bar{y}].$$

The upper limit may be finite (e.g., the support function associated with the constraint $y \leq \bar{y}$) or infinite. The meeting point between the linear and the nonlinear part is denoted y^c which is uniquely defined by

$$ty^c = \tilde{g}(y^c).$$

In view of the convexity of g, we have $\tilde{g}'(y^c) \geq t$. We assume that $\tilde{g}$ is continuously differentiable on the interior of its domain. Note that $\operatorname{dom} g = \operatorname{dom} \tilde{g}$. We name the function (5.20) a *compound congestion* function.

Let us compute the negative conjugate of the compound congestion function. Let

$$y(u) = \arg \min_{y \in \operatorname{dom} g} \{g(y) - \langle u, y \rangle\},$$

whenever the minimum occurs in $\operatorname{dom} \tilde{g}$. Simple calculation yields

$$-g_*(u) = \begin{cases} (t-u)y^c, & \text{with } y(u) = y^c, \qquad \text{if } t \leq u < \tilde{g}'(y^c), \\ -\tilde{g}_*(u), & \text{with } y(u) = [g']^{-1}(u), \qquad \text{if } \tilde{g}'(y^c) \leq u \leq g'(\bar{y}). \end{cases}$$

It follows that $\operatorname{dom} g_* \subset \{u \geq t\}$. In other words, we can add the constraint $u \geq t$ in the maximization of the Lagrangian dual function.

Finally, $g_*(u)$ is differentiable on the interior of $\operatorname{dom} g_* \subset \{t < u < g'(\bar{y})\}$ and

$$g'_*(u) = y(u).$$

Figure 5.4 and Figure 5.5 display the plot of compound functions and their negative conjugate in the case of Kleinrock and BPR functions, respectively. Note that the linear function (5.14) is a special case of compound congestion function without nonlinear part and with $y^c = c$.

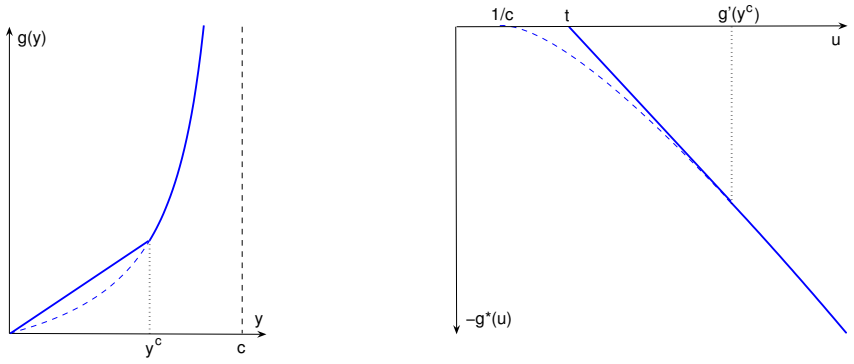

Figure 5.4: Compound Kleinrock function and its conjugate.

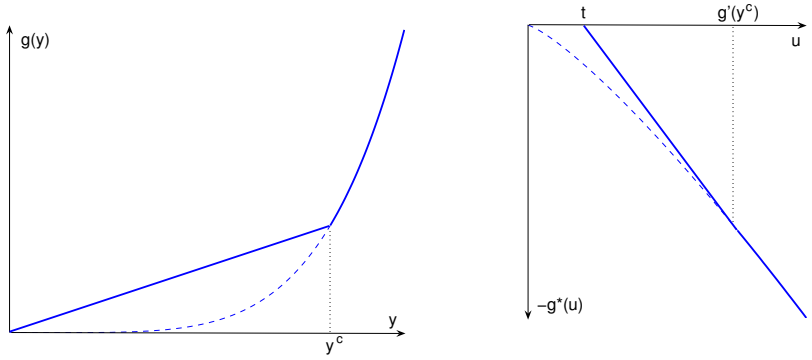

Figure 5.5: Compound BPR function and its conjugate.

5.5 Active set strategy on compound functions

In this section, we present a strategy to reduce the space of the Lagrangian problem when using compound congestion functions.

5.5.1 Motivations

Our interest for the class of compound congestion functions

$$g_a(y_a) = \max_{y_a \in \operatorname{dom} \tilde{g}_a} \{t_a y_a, \tilde{g}_a(y_a)\}, \quad a \in \mathcal{A},$$

which were introduced in the previous section, has been triggered by the observation that multicommodity flow problems with linear congestion functions and no capacity constraint are extremely easy to solve. Those problems are separable in the commodities, and for each commodity the minimization problem boils down to a shortest path calculation. In the case of capacity constraints on the arcs, we observe in practice that the number of saturated arcs at the optimum is a small fraction of the total number of arcs. In other words, for a large majority of arcs, the total flow in the optimal solution is strictly less than the installed capacity. Consequently, unsaturated capacity constraints are unnecessary and the associated Lagrangian dual variables are null at the optimum. If this (large) set of null optimal dual variables were known in advance, one could perform a partial Lagrangian relaxation restricted to the saturated arcs. This would considerably reduce the dimension of the Lagrangian dual and make it much easier to solve. In practice, the set of saturated arcs at the optimum is not known, but can be dynamically estimated. A heuristic that dynamically estimates the sets of active and inactive arcs has proved successful for linear MCF, [13]. It has also been implemented within the framework of bundle method to solve the Lagrangian dual [49] by Frangioni and Gallo and in a primal partitioning method [94] by McBride. Both papers report significant speed-ups.

This procedure can be extended to nonlinear MCF problems with a compound congestion since, on those arcs where $y_a < y_a^c$, the function is linear. We notice that linear functions are a special case of compound functions in which y_a^c is the capacity. It can be further extended to standard nonlinear MCF problems if one approximates the objective function by a compound one (see Section 5.6). In that situation, the aim of an active strategy is to take advantage of low congestion on certain arcs.

5.5.2 Optimal partition

Consider the first order optimality conditions for the Lagrangian dual problem (5.2). In full generality, the condition stipulates that $u \geq u^l$ is optimal if there exists a nonnegative vector τ such that

$$\begin{aligned} \tau + \partial f_1(u) + \partial f_2(u) &\ni 0, && (5.21)\\ \langle \tau, u - u^l \rangle &= 0, && (5.22)\\ \tau \geq 0,\ u &\geq u^l, && (5.23) \end{aligned}$$

where $u^l = g'_+(0)$. An anti-subgradient of f_1 is of the form Mx with $x \in \mathcal{X}$, while for f_2 one can take $-y(u)$. Hence, (5.21) and (5.22) imply

$$Mx \leq y(u),$$

and

$$(Mx)_a < y_a(u_a) \ \Rightarrow u_a = u^l_a = (g_a)'_+(0). \tag{5.24}$$

A similar analysis can be performed on the primal side.

Our goal is to use condition (5.24) to find a set $\mathcal{A}_1^* \subset \mathcal{A}$ with the property that for a optimal primal-dual pair (τ^*, u^*)

$$\tau^*_a > 0 \text{ and } u^*_a = u^l_a, \qquad \forall a \in \mathcal{A}_1^*.$$

If the set $\mathcal{A}_1^*$ were known in advance, the variables u_a, $a \in \mathcal{A}_1^*$, could be fixed to their lower bound u^l_a. The original problem (5.2) would then boil down to a simpler problem in the variables u_a, $a \in \mathcal{A}_2^* = \mathcal{A} \setminus \mathcal{A}_1^*$. Note that $f_{2a}(u^l_a) = 0$; thus, the nonlinear term $f_{2a}(u_a)$, $a \in \mathcal{A}_1^*$, can be removed from this equivalent formulation. The above reasoning applies to any subset $\tilde{\mathcal{A}}_1 \subset \mathcal{A}_1^*$ and its complement $\tilde{\mathcal{A}}_2 \supset \mathcal{A}_2^*$.

In view of the above partition, we define (5.2) as the partial Lagrangian problem

$$\max\{f(u) \mid u_a = u^l_a,\ a \in \mathcal{A}_1^*; \quad u_a \geq u^l_a,\ a \in \mathcal{A}_2^*\}.$$

5.5.3 Active set strategy

In practice, the partition is not known in advance and the proposed space reduction technique cannot be straightforwardly implemented. An active set strategy aims to guess the partition. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the current estimate of $\mathcal{A}_1^*$ and $\mathcal{A}_2^*$ with $\mathcal{A}_1 \cup \mathcal{A}_2 = \mathcal{A}$. This partition is used to work in a dual space of reduced dimension, a powerful gimmick if the cardinality of $\mathcal{A}_2^*$ is

small. How one should build these approximation sets? The danger is to have an arc that moves from a set to its complement back and forth. We propose heuristic rules to avoid this bad behavior.

Assumption 4. *The iterative procedure has generated Mx^i, $i = 1, \dots k$, with $x^i \in \mathcal{X}$ and it is always possible to construct flows y^k that meet all the demands.*

We also recall that y^c is the meeting point between the linear and the nonlinear part. Assuming we are given a current partition of $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ into an active set and its complement, the rules that move elements between $\mathcal{A}_1$ and $\mathcal{A}_2$ are:

- An arc $a \in \mathcal{A}_1$ such that $y_a^k > y_a^c$ is moved into the active set $\mathcal{A}_2$.
- An arc $a \in \mathcal{A}_2$ such that $y_a^k \leq \gamma y_a^c$ is moved into the non active set $\mathcal{A}_1$.

Note that we introduce the parameter $\gamma < 1$ to increase the chances that an arc that is made inactive at some stage will not become active later in the process. In practice, we get $\gamma = 0.9$.

5.5.4 Literature overview

The use of an active set strategy in solving the linear multicommodity flow problem is not new. It has been implemented within the framework of bundle method to solve the Lagrangian dual [49] and in a primal partitioning method [94]. Let us describe shortly these strategies.

In [49], Frangioni and Gallo have implemented a *Lagrangean variables generation* strategy to work on a subset of Lagrangean multipliers, i.e., the active set. Every 10 iterations, the Lagrangean dual problem is solved with all variables and all inactive variables with a positive ascent direction are added in the active set. The authors observed that usually the partition keeps stable after few iterations. They also observed a reduction by up to a factor 5 of CPU time spent in the computation of the master program even on smaller instances.

In [94], McBride proposes a *basis reduction heuristic* to reduce the dimension of the working inverse in the primal partitioning method EMNET. During the process the working inverse handles only active constraints and makes computations on each pivot faster. The active constraints are the constraints closed to be saturated. A constraint is declared active when it becomes saturated or close to be saturated. The first set of active constraints is

initialized applying a resource-directive decomposition heuristic. In the experiments, the author observed a reduction of the size of the working inverse from 17% to 70%, and a reduction of the CPU time up to 40%.

We notice that in these implementations, there is no strategy to remove a dual variable or a constraint from the active set to the inactive set. In other words, it is not possible to reduce dynamically the working space.

5.6 Approximation scheme

To exploit the active set strategy on nonlinear objective functions, we propose an approximation scheme that replaces the original nonlinear congestion function $\tilde{g}_a$ by a compound one g_a. Performing this approximation leads to the situation described in the previous section and makes the application of the active set strategy possible. Note that the approximation error is easily controlled by an appropriate choice of the meeting point y_a^c in (5.20). The approximation error tends to zero when $t_a \downarrow (\tilde{g}_a)'_+(0)$. In Section 5.4, we gave the formula for the negative conjugate of the compound congestion function. We can also use this expression to compute the maximal error on the dual side.

Assume that the nonlinear function $\tilde{g}$ is approximate such that

$$g_a(y_a) = \max_{y_a \in \operatorname{dom} \tilde{g}_a} \{t_a y_a, \tilde{g}_a(y_a)\}, \tag{5.25}$$

where $t_a \geq \tilde{g}_{a+}'(0)$. The linear and the nonlinear parts meet at y_a^c. The error function induced by the linear approximation

$$e_a(y_a) = t_a y_a - \tilde{g}_a(y_a), \quad y_a \in [0, y_a^c].$$

Let $\hat{y}_a \in [0, y_a^c]$ be the point that maximizes the error e_a. Let $\mathcal{A}_1^*$ be the set of inactive arcs at the optimum. The error due to the approximation is bounded by $\sum_{a \in \mathcal{A}_1^*} e_a(\hat{y}_a)$. We want that this error to be lower than $\epsilon \tilde{g}^*$, where ϵ is the relative optimality gap and $\tilde{g}^*$ is the unknown optimal objective value. Furthermore we impose that $e_a(\hat{y}_a) = \mu$ be the same for all $a \in \mathcal{A}_1^*$. We estimate μ by

$$\mu = \frac{\hat{\epsilon} \tilde{g}^*}{|\mathcal{A}_1^*|}, \quad \text{with } \hat{\epsilon} < \epsilon. \tag{5.26}$$

Then, we compute t_a such that $e_a(\hat{y}_a) = \mu$. Unfortunately $\tilde{g}^*$ and $|\mathcal{A}_1^*|$ must be estimated. In the experiments, we take the parameter $\hat{\epsilon} = 10^{-6}$ and $|\mathcal{A}_1^*| = n/2$. The value of $\tilde{g}^*$ is chosen empirically depending on the class of problems.

In the experiments, we approximate the Kleinrock delay function and the BPR congestion function by compound functions and use an active set strategy. The algorithm generates an ϵ-optimal primal-dual solution (y^*, u^*) for the compound function. The primal-dual pair can also be used to measure the relative optimality gap with the original functions $\tilde{g}$ and $\tilde{g}_*$. This gap depends on the quality of the approximation by the compound function and thus may be larger than ϵ.

5.7 Concluding remarks

In this chapter, we have pointed out that the Lagrangian dual problem is a sum of two functions. The first one is smooth and can be given in closed form. The second function is nonsmooth and may be described by polyhedral approximations. In the next chapter, we shall propose an ACCPM implementation which exploits the second order information of the smooth function.

In the second part of the chapter, we have presented an active set strategy to reduce the dual working space when using compound congestion functions. In practice, this strategy appears to be significantly efficient (see chapters on experiments).

Finally, we have proposed a approximation scheme motivated by compound functions in order to apply active set strategy on nonlinear functions. The present study suggests that possible further improvements could be achieved using the approximation/active set approach with a different linearization scheme for the cost function. Conceptually, this linearization could be performed around points that are dynamically chosen to lead more efficient approximations. This will be the object of further researches.

Chapter 6

Constrained analytic center cutting plane method

Contents

In this chapter, we present the analytic center cutting plane method, in short ACCPM, and some recent enhancements that include a proximal term and a logarithmic barrier on the epigraph of the smooth component of the objective function.

ACCPM is convenient to handle a class of convex optimization problems in which the information pertaining to the function to be minimized and/or to the feasible set takes the form of an outer linear approximation revealed by an *oracle*. By oracle, we mean a black-box scheme that returns appropriate information on the problem at so-called query points. Traditionally, the cutting plane method builds iteratively a polyhedral approximation of the problem, called *localization set*. The approximation is improved until a optimal solution is found. A main issue is to decide where to query the oracle in order to improve a current polyhedral approximation. ACCPM selects the analytic

center of the interior of this localization set, that is, the point that minimizes the logarithmic barrier function on that set, augmented with a proximal term. This choice is efficient since it usually requires relatively few query points to achieve an accurate approximation of an optimal solution.

The new approach also exploits the property that the objective function may have a smooth component with second order derivatives readily available in closed form. We use in the definition of the localization set a direct representation of the epigraph of the smooth component as a fixed nonlinear constraint. Then this fixed constraint is endowed with a self-concordant augmented barrier in the computation of the analytic center. This approach is similar to [100] but our implementation does not use the embedding into a projective space.

The chapter is organized as follows. The first section describes the general cutting plane approach. In the second section, the focus is on the computation of the analytic centers while, in the third one, we present the Newton method used to compute the analytic centers. Section four introduces some advance technic implementations that make ACCPM more efficient. Finally, the last section is devoted to implementation details.

6.1 Cutting plane method

We consider the general problem of the form

$$\min\{f(u) = f_1(u) + f_2(u) \mid u \in U_1 \cap U_2\}, \tag{6.1}$$

in which $U_i \subset \mathbb{R}^m,\ i = 1, 2,\ f_1 : \mathbb{R}^m \to \mathbb{R}$ is a convex nonsmooth function and $f_2 : \mathbb{R}^m \to \mathbb{R}$ is a convex self-concordant function. We further assume that U_2 is endowed with a self-concordant barrier. We shall consider two main possibilities for U_2. Either $U_2 = \mathbb{R}^m$, or,

$$U_2 = \{u \mid g_i(u) \leq 0,\ i = 1, \ldots, r\}.$$

In general, U_2 includes simple box constraints $\beta_l \leq u \leq \beta_u$ or a ball constraint $||u - u^c|| \leq 0$.

Finally, the nonsmooth function f_1 often is the positively weighted sum of p nonsmooth functions

$$f_1(u) = \sum_{i=1}^{p} \pi_i f_{1i}(u). \tag{6.2}$$

This property can be exploited in the solution method.

6.1.1 First and second order oracles

We assume that f_1 and U_1 are revealed by a *first order oracle*, while f_2 is accessed through a *second-order oracle*.

Definition 5. *A first-order oracle for problem (6.1) is a black box procedure with the following property. When queried at $\bar{u}$, the oracle returns 1 or 2.*

1. *$\bar{u} \notin U_1$ and (a, γ) is such that $a^T u - \gamma \leq 0, \forall u \in U_1$ (feasibility cut). In that case, we set $f_1(u) = +\infty$.*

2. *$\bar{u} \in U_1$ and (a, γ) is such that $a^T u - \gamma \leq f_1(u), \forall u \in U_1$ (optimality cut).*

Note that, in general, parameters defining the optimality cut are such that $a \in \partial f_1(\bar{u})$ and $\gamma = a^T \bar{u} - f_1(\bar{u})$. But this is not necessarily so. The cut may have no intersection with the epigraph set (i.e., may be situated strictly below that set).

Definition 6. *A second-order oracle for problem (6.1) is a black-box procedure with the following property. When queried at $\bar{u}$, the oracle returns the function value and the first and second derivatives of $f_2(u)$ at $u = \bar{u}$.*

In Definition 5, we assume that the first order oracle returns only one cutting plane when queried to a given point $\bar{u}$. This assumption is too restrictive. The first order oracle may produce multiple feasibility or optimality cuts. We distinguish the two main situations:

- Consider the following feasible set

$$U_1 = \{u \mid h_i(u) \leq 0,\ i = 1, \ldots, k\}.$$

  It is easy to see that the query point $\bar{u}$ can violate few constraints h_i. Thus, the oracle may compute multiple feasibility cuts, i.e., as many as the number of violated constraints.

- Assume now that the objective function f_1 is defined by (6.2). In that situation, the first order oracle may exploit the disaggregation of the objective function and returns p optimality cuts, i.e., one for each component f_{1i}. This oracle is conveniently named *disaggregate oracle*. In contrast, the standard oracle which returns only one optimality cut will be named *aggregate oracle*.

6.1.2 The localization set

Let z and ζ be the epigraph variables associated, respectively, to the epigraph of f_1 and f_2. The epigraph of the function f is the set defined by

$$\{(u, z, \zeta) \mid \pi^T z \geq f_1(u),\ \zeta \geq f_2(u)\}.$$

Using this property, problem (6.1) can also be written as

$$\begin{aligned} \min \quad & \pi^T z + \zeta && && (6.3a)\\ & f_{1i}(u) - z_i \leq 0, && i = 1, \ldots, p, && (6.3b)\\ & f_2(u) - \zeta \leq 0, && && (6.3c)\\ & u \in U_1 \cap U_2. && && (6.3d) \end{aligned}$$

The first order oracle is used to build a polyhedral approximation of the epigraph of f_1. Suppose the oracle has been queried at $u^i,\ i = 1, \ldots, k$, and has returned n feasibility and/or optimality cuts associated with those points. The corresponding inequalities, called cutting planes, are collected in

$$A^T u - E^T z \leq \Gamma.$$

In that definition, the subgradients a of the function f_1 form the matrix A, the constants γ are collected in the vector Γ, while E is a binary matrix that is constructed as follows. If the objective f_1 is treated in an aggregate mode, then E is a binary row vector. An entry one in E indicates that the z variable is present in the cut, implying that the cut is an *optimality cut*. In contrast, a zero indicates that the cut is a *feasibility cut*. If the objective f_1 is disaggregated into p components, E is a matrix of the form

$$E = \begin{pmatrix} 1 \ldots 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 \ldots 1 & \cdots & 0 & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \ldots 1 & 0 & \cdots & 0 \end{pmatrix}.$$

A row i of E corresponds to a variable z_i and each column corresponds to a cut. An entry one in row i and column j indicates that the cut j is an optimality cut for $f_{1i}(u)$. If column j is a null vector, then cut j is a feasibility cut.

Let $\bar{\theta}$ be the best recorded value such that

$$\bar{\theta} = \min_{i \leq k} \{f_1(u^i) + f_2(u^i)\}, \tag{6.4}$$

we can defined the so-called *hat cut*

$$\pi^T z + \zeta \leq \bar{\theta}. \tag{6.5}$$

In view of the above definitions, we can define the localization set $\mathcal{L}_k$ as

$$\mathcal{L}_k = \left\{(u, z, \zeta) \mid A^T u - E^T z \leq \Gamma,\ f_2(u) \leq \zeta,\ \pi^T z + \zeta \leq \bar{\theta},\ u \in U_2\right\}, \tag{6.6}$$

which is a subset of an outer approximation of the epigraph of f that contains all optimal pairs $(u^*, f(u^*))$.

Assumption 5. *The set of localization $\mathcal{L}_k$ has a non-empty interior.*

Remark 8. *Assumption 5 is met if $\{dom\, f_1 \cup dom\, f_2 \cup U_2\} \neq \emptyset$ and $\bar{\theta} \geq \theta^*$. The first condition holds in general. To enforce the second condition, we replace $\bar{\theta}$ by $\bar{\theta} + \nu$, where ν is a small fraction of the optimality tolerance ϵ.*

6.1.3 Bounds on the objective function

In this subsection, we introduce the upper and lower bounds for the objective function. These bounds permits a measure of progress to optimality.

Upper bound

By duality, any feasible solution of (6.1) provides a upper bound. Taking the values returned by the two oracles at the successive query points, we obtain the upper bound

$$\bar{\theta} = \min_{i \leq k} \{f_1(u^i) + f_2(u^i)\}. \tag{6.7}$$

Lower bound

We now explain a way to generate a lower bound. The first step in the derivation of the lower bound consists in introducing the perturbed function $f(u) - r^T u$, where r is a vector to be specified later. The second step is to replace the non-smooth function $f_1(u)$ by its current polyhedral approximation. This is done by replacing $f_1(u)$ by $\pi^T z$ under the constraints $A^T u - E^T z \leq \Gamma$. We have thus the bounding inequality

$$f(u) - r^T u \geq \min_{u,z} \{\pi^T z + f_2(u) - r^T u \mid A^T u - E^T z \leq \Gamma\}.$$

In view of the convexity of f_2, we may write

$$\begin{aligned} f(u) - r^T u &\geq f_2(u^c) - f_2'(u^c)^T u^c + \\ &\quad \min_{u,z}\{\pi^T z + f_2'(u^c)u - r^T u \mid A^T u - E^T z \leq \Gamma\}, \end{aligned}$$

where u^c is a point of choice (e.g., an approximate analytic center). By duality we obtain

$$\begin{aligned} f(u) - r^T u &\geq f_2(u^c) - f_2'(u^c)^T u^c + \\ &\quad \min_{u,z} \max_{x\geq 0}\{(f_2'(u^c) + Ax)^T u + (\pi - Ex)^T z - \Gamma^T x - r^T u\}, \\ &= f_2(u^c) - f_2'(u^c)^T u^c + \max_{x\geq 0}\{ -\Gamma^T x \\ &\quad + \min_{u,z}\left[(f_2'(u^c) + Ax - r)^T u + (\pi - Ex)^T z\right]\Big\}. \end{aligned} \tag{6.8}$$

If $x \geq 0$ is such that

$$Ex = \pi, \tag{6.9}$$

and

$$r = f_2'(u^c) + Ax, \tag{6.10}$$

then the min operand is null and

$$f(u) \geq f_2(u^c) - f_2'(u^c)^T u^c + r^T u - \Gamma^T x.$$

Some methods, such as Kelley's method, the bundle method or the subgradient methods, are known to generate a x satisfying (6.9) and (6.10) in their process. It is also the case for ACCPM at the end of the iterations that compute the proximal analytic center. One may also expect r to be small. We obtain the bound for the optimal objective function value by

$$\begin{aligned} f(u^*) &\geq f_2(u^c) - f_2'(u^c)^T u^c - \Gamma^T x^c + r^T u^*, \\ &\geq f_2(u^c) - f_2'(u^c)^T u^c - \Gamma^T x^c + r^T(u^* - u^c) + r^T u^c, \\ &\geq f_2(u^c) - f_2'(u^c)^T u^c + r^T u^c - \Gamma^T x^c - ||r||\delta. \end{aligned} \tag{6.11}$$

The last inequality follows from Cauchy-Schwartz and $\delta \geq ||u^* - u^c||$ is an upper bound on the distance of the current point u^c to the optimal set. Finding a good value for δ cannot be done on theoretical grounds. It is essentially problem dependent. In practice, we obtained good results by taking the "empirical" value $\delta = 5 \times ||u^c - \bar{u}||$.

If the variable u is constrained to be nonnegative in (6.1), we can further improve the computation of the lower bound by taking $r = -\min\{0, f_2'(u^c) + Ax^c\}$, where the min operand is taken component-wise. In that case, the coefficient of u in the inner minimization is always nonnegative and $(f_2'(u^c) + Ax - r)^T u = 0$ at the solution of (6.8). This remark is particularly useful when $r = 0$. Then we obtain the exact lower bound $f_2(u^c) - f_2'(u^c)^T u^c - \Gamma^T x^c$.

Remark 9. *The above analysis shows that one may expect primal feasible solution and exact lower bound for MCF problem, when $r = 0$, but it does not provide convergence theoretical proof. Nevertheless, in practice we always observe both $r = 0$ after a few iterations and convergence of the lower bound to the optimal solution.*

6.1.4 Termination criterion

The standard termination criterion is a small enough relative optimality gap

$$(\bar{\theta} - \underline{\theta})/\max(\underline{\theta}, 1) \leq \epsilon, \tag{6.12}$$

where $\bar{\theta}$ and $\underline{\theta}$ are respectively the best computed upper and lower bounds for the objective function f. In our experiments we use $\epsilon = 10^{-5}$.

6.1.5 The algorithm

In the proposed version of ACCPM, the query point is an approximate proximal analytic center of the localization set defined as the intersection of cutting planes and a fixed cutting surface. For the sake of clarity, we sketch the basic step, or *outer iteration*, of a generic cutting plane method.

Algorithm 8: Outer iteration of constrained ACCPM

1. Select a query point in the localization set (Inner iteration).
2. Send the query point to the first order oracle and get back optimality/feasible cuts.
3. Send the query point to the second order oracle to compute the objective function f_2.
4. Update the lower and upper bounds and the localization set.
5. Test termination.

6.2 Inner iterations

It is well-known that efficient methods for non differentiable convex optimization rely on some regularization scheme to select the query point. We discuss here such a scheme; it is based on the concept of proximal analytic center which is defined as the unique minimizer of a weighted logarithmic barrier for the localization set, augmented with a proximal term.

6.2.1 Barrier for the localization set

We associate with constraints of the localization set a standard weighted logarithmic barrier. We have

$$F(\bar{s}) = -w_0 \log s_0 - \sum_{i=1}^{n} w_i \log s_i - \log \sigma, \tag{6.13}$$

where $\bar{s} = (s_0, s, \sigma) > 0$ is defined by

$$\begin{aligned} s_0 &= \bar{\theta} - (\pi^T z + \zeta), \\ s_i &= \Gamma_i - (A^T u - E^T z)_i, \quad i = \{1, \ldots, n\}, \\ \sigma &= \zeta - f_2(u). \end{aligned}$$

We also assume that the set U_2 is endowed with a self-concordant barrier $H(u)$. In most applications the set U_2 is defined by simple constraints. In this work we consider the two main situations:

Ball constraint This constraint restricts the points to be in a ball of radius R centered at u^r. The barrier

$$H(u) = -\log(R^2 - ||u - u^r||^2),$$

is self-concordant with self-concordant parameter $\nu = 1$.

Box constraints The simple box constraints take the form $\beta_l \leq u \leq \beta_u$. The associated barrier is

$$H(u) = -\sum_{i=1}^{m}(\log(u - \beta_l)_i + \log(\beta_u - u)_i).$$

6.2.2 Proximal term

The barrier function is augmented with a proximal term to yield the augmented barrier

$$\frac{1}{2}(u - \bar{u})^T Q(u - \bar{u}) + F(\bar{s}) + H(u),$$

where Q is a positive definite matrix. In practice, we get $Q = \rho I$. The proximal reference point $\bar{u}$ and the proximal coefficient ρ are arbitrary. In practice, the initial proximal reference point $\bar{u}$ is the first query point. Thereafter, $\bar{u}$ is chosen to be the query point that achieves the best recorded value $\bar{\theta}$, i.e.,

$$\bar{u} = \arg\min_{i \leq k}\{f_1(u^i) + f_2(u^i)\}.$$

6.2.3 Proximal analytic centers

The proximal analytic center method defines the next query point to be the u component of the solution (u, z, ζ) to the minimization problem

$$\min \quad \frac{\rho}{2}||u - \bar{u}||^2 - w_0 \log s_0 - \sum_{i=1}^{n} w_i \log s_i - \log \sigma + H(u) \quad (6.14a)$$

$$s_0 = \bar{\theta} - (\pi^T z + \zeta) \geq 0, \quad (6.14b)$$

$$s_i = \Gamma_i - (A^T u - E^T z)_i, \quad i = \{1, \ldots, n\}, \quad (6.14c)$$

$$\sigma = \zeta - f_2(u) \geq 0. \quad (6.14d)$$

Theorem 4. *Under Assumption 5, problem (6.14) achieves its minimum value at a unique point.*

Proof. Let (u, z, ζ) belong to the interior of localization set $\mathcal{L}_k$; thus $\bar{s} = (s_0, s > 0, \sigma) > 0$. In view of the simple inequality $-\log x \geq 1 - x$, we have

$$F(\bar{s}) \geq -w_{max} s_0 - \sum_{i=1}^{n} w_{max} s_i - w_{max}\sigma + 2w_{max} + n \sum_{i=1}^{n} w_{max},$$

with $w_{max} = \max(w_0, w_1, \ldots, w_n, 1)$. Since $s_0 \geq 0$ and $\sigma \geq 0$ and in view of the convexity of f_2, we have

$$\begin{aligned} s_0 + \sum_{i=1}^{n} s_i + \sigma &\leq \sum_{i=1}^{n} (s_0 + s_i + \sigma), \\ &\leq n\bar{\theta} + \sum_{i=1}^{n} (\Gamma_i - (A_i)^T u) - n f_2(u), \\ &\leq n\bar{\theta} + \sum_{i=1}^{n} (\Gamma_i - (A_i)^T u) - n(f_2(\hat{u}) + f_2'(\hat{u})^T (u - \hat{u})). \end{aligned}$$

Finally, by convexity of $H(u)$, we have

$$\frac{\rho}{2}||u - \bar{u}||^2 + H(u) \geq \frac{\rho}{2}||u - \bar{u}||^2 + H(\hat{u}) + H'(\hat{u})^T (u - \hat{u}).$$

It follows that the objective of (6.14) is bounded from below on its feasible set by a strongly quadratic convex function. This bounding quadratic function has bounded level sets. Thus, the projection of a level set of (6.14) on u, which is contained in some level set of the bounding function, is also bounded. We can easily show that (6.14b)–(6.14d) imply that z and ζ are also bounded from below and above on the level set of (6.14). Therefore, problem (6.14) achieves its minimum, and by strict convexity, this minimum is unique. □

6.2.4 First order optimality conditions

The first order optimality conditions for problem (6.14) are

$$\begin{aligned} \rho(u - \underline{u}) + Aws^{-1} + f_2'(u)\sigma^{-1} + H'(u) &= 0, && \text{(6.15a)} \\ -E^T w s^{-1} + w_0 \pi^T s_0^{-1} &= 0, && \text{(6.15b)} \\ w_0 s_0^{-1} - \sigma^{-1} &= 0, && \text{(6.15c)} \\ s - \Gamma - E^T z + A^T u &= 0, && \text{(6.15d)} \\ s_0 - \bar{\theta} + \pi^T z + \zeta &= 0, && \text{(6.15e)} \\ \sigma - \zeta + f_2(u) &= 0. && \text{(6.15f)} \end{aligned}$$

It is possible to interpret $x_0 = w_0 s_0^{-1}$, $x_i = w_i s_i^{-1}$, $i = 1, \ldots n$, and $\xi = \sigma^{-1}$ as "primal" variables. We then have the complementary condition $xs = w$ and $\xi\sigma = 1$. The vector $x = ws^{-1}$ is used in (6.11) to derive a lower bound for the optimal solution.

To write the generalized analytic center problem in a more condensed format, we introduce the variable $v = (u, z, \zeta)$ and collect all the constraints (6.14b)-(6.14d) into $h(v) \leq 0$. We find it convenient to formulate the constraints as

$$h(v) + \bar{s} = 0,\ \bar{s} = (s_0, s_1, \cdots, s_n, \sigma) \geq 0.$$

Finally, writing

$$K(u) = \frac{1}{2}(u - \bar{u})^T Q(u - \bar{u}) + H(u),$$

we formulate the generalized analytic center problem as

$$v^c = (u^c, z^c, \zeta^c) = \arg\min_{v,s}\{K(u) + F(\bar{s}) \mid h(v) + \bar{s} = 0,\ \bar{s} > 0\}. \tag{6.16}$$

6.2.5 Damped Newton step

The algorithm that computes the analytic center is a damped Newton method applied to (6.15). At each inner iteration, the method computes a Newton direction $(du, dz, d\zeta, ds, ds_0, d\sigma)$. The method is described in the next section. Since (6.15f) is nonlinear, a full Newton step does not yield a feasible point with respect to (6.15f). Thus, we use the following empirical rule to compute the step length α_{step}. Let $0 < \gamma < 1$ be a fixed parameter and

$$\alpha_{max} = \max\{\alpha \mid s_0 + \alpha ds_0 > 0,\ s + \alpha ds > 0,\ \sigma + \alpha d\sigma > 0,\ u \in U_2\},$$

the step length is

$$\alpha_{step} = \min(1,\ \gamma\alpha_{max}). \tag{6.17}$$

6.2.6 The algorithm

To summarize, a basic step of the Newton iteration, or *inner iteration*, is described by Algorithm 9.

Remark 10. *The computation in the inner iteration uses second order derivatives of f_2. Since the derivatives change at each iteration, the inner iteration must have access to the second order oracle. In pure cutting plane methods, the inner iteration does not interact with the oracle.*

Algorithm 9: Inner iteration of Newton's method

1. Send the current point to the second order oracle to compute the objective function f_2 and its first and second derivatives.
2. Compute the Newton step $(du, dz, d\zeta, ds, ds_0, d\sigma)$ by (6.23).
3. Compute a step length by (6.17) to update $(u, z, \zeta, s, s_0, \sigma)$.
4. Test termination.

6.3 Newton's method

In this section, we use the following notation. Given a vector $s > 0$, S is the diagonal matrix whose main diagonal is s. We also use $s^{-1} = S^{-1}e$ to denote the vector whose coordinates are the inverse of the coordinates of s. Similarly, $s^{-2} = S^{-2}e$.

6.3.1 Definition

The aim is to minimize $G(v) = K(u) + F(-h(v))$ where $v = (u, z, \zeta)$. The method of choice is Newton's method. Let us first briefly review the case of a feasible Newton method. The Newton direction is

$$dv = -[G''(v)]^{-1}G'(v).$$

The variant of Newton's method for computing the proximal generalized analytic center consists in taking damped steps to preserve feasibility of u, z and ζ. The aim is to achieve a sufficient decrease of G, until the area of quadratic convergence is hit. From then on, the method takes full Newton steps, with no line-search. We recall that the sufficient condition for guaranteed quadratic convergence is

$$\langle [G''(v)]^{-1}G'(v), G'(v)\rangle = \langle -dv, G'(v)\rangle < 1. \tag{6.18}$$

The left-hand side of the inequality is a proximity measure. When the proximity is below the unit threshold, then the point $v + dv$ is feasible and quadratically closer to the generalized analytic center (smaller proximity measure). The stopping criterion is a threshold value $\eta < 1$ on the proximity. To enforce this proximity at $v + dv$, it suffices that the proximity at v be less than $\sqrt{\eta}$.

The stopping criterion (6.18) does not imply that $G'(v) = 0$, but most likely $G'(v)$ will be close to zero and $G'(v + dv)$ even closer.

6.3.2 Infeasible start

Problem (6.16) raises the issue of feasibility. In cutting plane schemes, the new constraints exclude the current iterate from the new localization set. There is no direct way to retrieve feasibility if the cuts are deep. We propose an infeasible start Newton method, which aims to achieve feasibility and optimality simultaneously.

Let us explicit the first and second derivatives.

$$\begin{aligned} G'(v) &= K'(u) - \frac{\partial h}{\partial v} F'(-h(v)), \\ G''(v) &= K''(u) + \frac{\partial h}{\partial v} F''(-h(v)) \frac{\partial h}{\partial v}^T. \end{aligned}$$

Using the intermediate slack variable $\bar{s} = (s_0, s, \sigma)$, we obtain for the first optimality conditions for (6.16)

$$\begin{aligned} K'(u) - \frac{\partial h}{\partial v} F'(\bar{s}) &= 0, \\ h(v) + \bar{s} &= 0. \end{aligned}$$

In the course of the optimization process, those equations are never satisfied. However, we assume that $\bar{s} > 0$, and we introduce the residual r and write

$$K'(u) - \frac{\partial h}{\partial v} F'(\bar{s}) = r_d, \tag{6.19}$$

$$h(v) + \bar{s} = r_p. \tag{6.20}$$

The Newton direction in the $(dv, d\bar{s})$ space with $d\bar{s} = (ds_0, ds, d\sigma)$ is given by

$$K''(u)du \quad - \frac{\partial h}{\partial v} F''(\bar{s}) d\bar{s} = -r_d, \tag{6.21}$$

$$\frac{\partial h}{\partial v}^T dv \qquad + d\bar{s} \qquad = -r_p. \tag{6.22}$$

6.3.3 Newton's direction

Prior to discussing ways of solving (6.21)-(6.22), we explicit the components in the equations (6.19) to (6.22). We have

$$K'(u) = \begin{pmatrix} Q(u - \bar{u}) + H'(u) + f_2'(u) \\ 0 \\ 0 \end{pmatrix},$$

$$K''(u) = \begin{pmatrix} Q + H''(u) + f_2''(u) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

and

$$\frac{\partial h}{\partial v} = \begin{pmatrix} 0 & A & f_2'(u) \\ \pi & -E & 0 \\ -1 & 0 & 1 \end{pmatrix}.$$

Finally we have

$$F'(\bar{s}) = \bar{w}\bar{s}^{-1},$$

with $\bar{w} = (w_0, w, \omega)$, and

$$F''(s) = \bar{W}\bar{S}^{-2}.$$

The algorithm that computes the analytic center is a damped Newton method applied to (6.15). To write down the formulae, we introduce the residuals

$$\begin{aligned} r_u &= -(\rho(u - \underline{u}) + Aws^{-1} + f_2'(u)\sigma^{-1} + H'(u)), \\ r_z &= -(-E^T ws^{-1} + w_0 \pi^T s_0^{-1}), \\ r_\zeta &= -(w_0 s_0^{-1} - \sigma^{-1}), \\ r_s &= -(s - \Gamma - E^T z + A^T u), \\ r_{s_0} &= -(s_0 - \bar{\theta} + \pi^T z + \zeta), \\ r_\sigma &= -(\sigma - \zeta + f_2(u)). \end{aligned}$$

Let $r_v = (r_u, r_z, r_\zeta, r_s, r_{s_0}, r_\sigma)$, the Newton direction $dv = (du, dz, d\zeta, ds, ds_0, d\sigma)$ associated to (6.15) is given by

$$N dv = r_v, \tag{6.23}$$

with

$$N = \begin{pmatrix} \Delta & 0 & 0 & -A\Lambda & 0 & -f_2'(u)\sigma^{-2} \\ 0 & 0 & 0 & E^T\Lambda & -\pi w_0 s_0^{-2} & 0 \\ 0 & 0 & 0 & 0 & -w_0 s_0^{-2} & \sigma^{-2} \\ A^T & -E^T & 0 & I & 0 & 0 \\ 0 & \pi^T & 1 & 0 & 1 & 0 \\ f_2'(u)^T & 0 & -1 & 0 & 0 & 1 \end{pmatrix},$$

where

$$\Lambda = WS^{-2},$$

and

$$\Delta = \rho I + f_2(u)''\sigma^{-1} + H''(u).$$

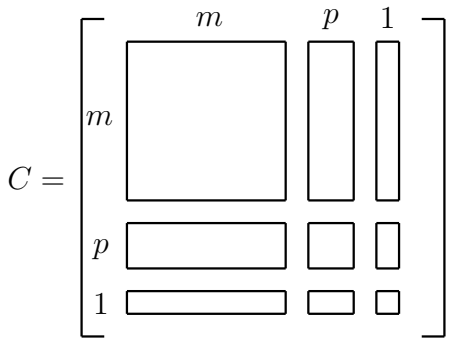

Figure 6.1: Structure of C when $p \leq m$.

6.3.4 Solving the Newton system

After pivoting on the SE diagonal block of the matrix, we obtain the equivalent system:

$$C \begin{pmatrix} du \\ dz \\ d\zeta \end{pmatrix} = \begin{pmatrix} r_u + A\Lambda r_s + \sigma^{-2} f_2'(u) r_\sigma \\ r_z - E\Lambda r_s + \pi w_0 s_0^{-2} r_{s_0} \\ r_\zeta + w_0 s_0^{-2} r_{s_0} - \sigma^{-2} r_\sigma \end{pmatrix}, \tag{6.24}$$

where

$$C = \begin{pmatrix} M & -A\Lambda E^T & -\sigma^{-2} f_2'(u) \\ -E\Lambda A^T & E\Lambda E^T + \pi w_0 s_0^{-2} \pi^T & \pi w_0 s_0^{-2} \\ -\sigma^{-2} f_2'(u)^T & \pi^T w_0 s_0^{-2} & w_0 s_0^{-2} + \sigma^{-2} \end{pmatrix}, \tag{6.25}$$

and

$$M = \Delta + A\Lambda A^T + \sigma^{-2} f_2'(u) f_2'(u)^T.$$

In the ACCPM computations, most of the time is spent in solving (6.24) and thus (6.23). It is then essential to exploit the structure of the problem (6.24) to implement efficient strategies. We propose alternative ways to solve (6.24), according to the relative dimensions n, and m of the variables, respectively, s and u. The block structure of the matrix C in (6.24) is displayed on Figure 6.1. We distinguish the two following ways to solve (6.24).

Case 1: $n \leq m$

When the number of generated cuts n is smaller than the dimension of the problem m, we suggest pivoting on M to solve (6.24). We obtain the following equivalent system

$$\begin{pmatrix} T & R \\ R^T & W \end{pmatrix} \begin{pmatrix} dz \\ d\zeta \end{pmatrix} = \begin{pmatrix} \varphi_z \\ \varphi_\zeta \end{pmatrix}, \tag{6.26}$$

with

$$\begin{aligned} T &= E\Lambda E^T + w_0 s_0^{-2}\pi\pi^T - E\Lambda A^T M^{-1} A\Lambda E^T, \\ R &= w_0 s_0^{-2}\pi - E\Lambda A^T M^{-1}\sigma^{-2} f_2'(u), \\ W &= w_0 s_0^{-2}\sigma^{-2} + \sigma^{-2} - f_2'(u)^T M^{-1}\sigma^{-2} f_2'(u), \\ \varphi_z &= r_z - E\Lambda r_s + w_0 s_0^{-2} r_{s_0}\pi + E\Lambda A^T M^{-1}(r_u + A\Lambda r_s + \sigma^{-2} r_\sigma f_2'(u)), \\ \varphi_\zeta &= r_\zeta + w_0 s_0^{-2} r_{s_0} - \sigma^{-2} r_\sigma + \sigma^{-2} f_2'(u)^T M^{-1}(r_u + A\Lambda r_s + \sigma^{-2} r_\sigma f_2'(u)). \end{aligned}$$

To compute the inverse M^{-1}, we use the Shermann-Morrison formula. Indeed, the matrix

$$B = A\Lambda^{\frac{1}{2}} + \frac{1}{2}\sigma^{-1} f_2'(u),$$

has fewer columns than rows and we may write

$$M^{-1} = \Delta^{-1} - \Delta^{-1} B(I + B^T\Delta^{-1}B)^{-1} B^T\Delta^{-1}.$$

The inner matrix $(I + B^T\Delta^{-1}B)$ has dimension $n \times n$ which is quite smaller than the dimension of M.

Assuming that the inverse of Δ is easy to compute, the complexity of the strategy, due to the computation of $B^T\Delta^{-1}B$ in the Shermann-Morrison formula, is $\mathcal{O}(mn^2)$ flops. Other computations are matrix-vector products. We will see later that the computation of Δ^{-1} is not expensive.

Case 1: $n \geq m$

When the number of generated cuts n is large, the matrix C still has the structure given in Figure 6.1, but the strategy proposed in the previous case is not more advantageous. Then, we propose to solve (6.24) by factoring the matrix directly, without any preliminary block pivot. In that situation, the complexity, due to the computation of M, is $\mathcal{O}(m^2 n)$ flops.

Ball and box constraints

To compute efficiently the inverse of M in the Shermann-Morrison formula, the inverse of Δ has to be easy to perform. We recall that the matrix Δ is the sum of two components. The first one, $\rho I + f_2(u)''\sigma^{-1}$, is a simple

diagonal matrix, and the second one, $H''(u)$, is the second derivative of the barrier function $H(u)$. Thus, the inversion difficulty depends directly on the structure of H.

Let us consider first the standard box constraints $\beta_l \leq u \leq \beta_u$. Let

$$H(u) = -\sum_{i=1}^{m} \log(\beta_u - u)_i - \sum_{i=1}^{m} \log(u - \beta_l)_i.$$

The Hessian of the barrier is a diagonal matrix. The i-th entry on the diagonal is

$$H''_{ii}(u) = \frac{1}{(\beta_u - u)_i^2} + \frac{1}{(u - \beta_l)_i^2}.$$

The barrier function associated with the ball constraints $||u - u^r|| \leq R$ takes the form $H(u) = -\log(R^2 - ||u - u^r||^2)$ be the associated self-concordant barrier. Then

$$H''(u) = \frac{4}{(R^2 - ||u - u^r||^2)^2} \left(\frac{R^2 - ||u - u^r||^2}{2} I + uu^T \right).$$

In that case, we note that $H''(u)$ is the sum of a diagonal matrix and a matrix with a rank one correction. Thus, it is worth pointing out that the inverse of $\rho I + f_2(u)''\sigma^{-1} + H''(u)$ is easily computed in both cases.

6.3.5 Stopping criterion

Let $g(x)$ be a self-concordant function with bounded level sets. Then it is well-known [98] that the Newton method converges quadratically in the neighborhood

$$\left\{ x \mid \langle -[g''(x)]^{-1} g'(x), g'(x) \rangle \leq \eta < \frac{3 - \sqrt{5}}{2} \right\}, \tag{6.27}$$

and that a damped Newton method converges to that neighborhood in a number of iterations that is polynomial.

Nevertheless, since it is not essential in our solution method to compute an exact analytic center, we use the termination criteria $\eta = 0.99$ that is looser than (6.27). The solution is then a η-approximate analytic center.

6.3.6 Convergence of the inner iterations

In this section, we first recall the main theoretical convergence results of Newton's method to compute the analytic centers in general cutting plane frameworks. We do not provide extensions of these results to the proposed Constrained ACCPM that includes a proximal term and a second order oracle. Nevertheless, in the next part we show that the objective function (6.14a) to be minimized is a self-concordant function in MCF context (see the proof in appendix A). Thus, we can apply useful properties of self-concordant functions to reach conclusion of convergence for the Newton method.

Convergence results for the computation of analytic centers

In [59], the authors give results of convergence for a method with linear cuts and box constraints in the framework of feasibility problems. The analytic center is computed as the unique solution of the minimization problem

$$\min\{-\sum_{i=1}^{m}\log s_i \;\mid\; A^T u + s \leq \Gamma, s > 0\}.$$

Let η be a centering parameter. The number of damped Newton steps to generate a η-approximation analytic center is bounded by $\frac{g(\eta)}{\eta-\log(1+\eta)}$, with

$$g(\eta) = -(1-\eta)()\sigma_1 + \sigma_2) + \log\frac{(1+\sigma_2)\sigma_2}{(1-\sigma_1)\sigma_1},$$

where

$$\sigma_1 = \frac{-(1+\eta)+\sqrt{(1-\eta)^2+4}}{2(1-\eta)},$$

and

$$\sigma_2 = \frac{1+\eta+\sqrt{(1-\eta)^2+4}}{2(1-\eta)}.$$

Convergence of Newton's method has been also analyzed, in [101], for a homogeneous version of the analytic center cutting plane method. In this version, the original problem is embedded in a projective space, i.e., in a cone K, and a quadratic term is introduced to penalize query points with too large norm. Finally, the set K is assumed to be equipped with ν-normal barrier H. The analytic center is then the unique solution of the minimization problem

$$\min\{\frac{1}{2}||u||^2 + H(u) - \sum_{i=1}^{m}\log\langle f(u_i), u_i - u\rangle\}.$$

The authors show that $\mathcal{O}(1)$ damped Newton steps followed by $\mathcal{O}(\ln\ln(1/\eta))$ full Newton steps generate a η-approximation analytic center. The damped steps are performed to retrieve feasibility.

Minimization of self-concordant functions

Let us consider the general problem:

$$\min\{g(x) \mid x \in \operatorname{dom} g\}, \tag{6.28}$$

where g is a self concordant function on $\operatorname{dom} g$. Let denote

$$\lambda_g(x) = \langle -[g''(x)]^{-1} g'(x), g'(x) \rangle,$$

the local norm of the gradient $g'(x)$. The next theorem provides a sufficient condition to establish that a solution to (6.28) exists and is unique.

Theorem 5. *(Theorem 4.1.11 of [98])*
Let $\lambda_g(x) < 1$ for some $x \in \operatorname{dom} g$. Then the solution x_g^* of problem (6.28) exists and is unique.

Theorem 6. *(Theorem 4.1.14 of [98])*
Let $\lambda_g(x) < 1$ for some $x \in \operatorname{dom} g$. Then the point $x^+ = x - [g''(x)]^{-1} g'(x)$ belongs to $\operatorname{dom} g$. Moreover,

$$\lambda_g(x^+) \leq \left(\frac{\lambda_g(x)}{1 - \lambda_g(x)} \right)^2. \tag{6.29}$$

Theorem 7. *(Theorem 4.1.12 of [98])*
Let $x \in \operatorname{dom} g$ and $x^+ = x - \frac{1}{1+\lambda_g(x)} [g''(x)]^{-1} g'(x)$. We have

$$f(x^+) \leq f(x) - \omega(\lambda_g(x)).$$

The condition of Theorem 6 defines the region of quadratic convergence of standard Newton's method. By solving the inequality (6.29) we get that this region is defined by $\lambda_g(x) \leq \frac{3-\sqrt{5}}{2}$. From Theorem 7, damped Newton steps ensure fix decreases of the function to reach this region. Thus, Newton's method applied to an unconstrained minimization of a self concordant function is globally convergent and quadratically convergent in the region defined by $\lambda_g(x) \leq \frac{3-\sqrt{5}}{2}$.

Note that the previous development is relevant only with feasible starting point. In cutting plane schemes, the new constraints may exclude the current

iterate from the new localization set and there is no direct way to retrieve feasibility if the cuts are deep. Thus, convergence of the infeasible Newton method is not straightforward. Nevertheless, in practice our method always retrieves feasibility. Assuming such a behavior and since the objective function is self-concordant in the multicommodity flow context (see the proof in appendix A), we conclude that the Newton method converges to a unique solution in the present situation.

6.4 Advanced topics

In this section, we present some advanced topics that have been implemented into ACCPM during the thesis. Some of these topics appears to be useful to make ACCPM more efficient on the multicommodity flow problems.

6.4.1 Acceleration techniques

It is well-known that column generation techniques are adversely affected by the total number of generated columns. This is particularly true with ACCPM, since the Newton iterations in the computation of the analytic center have a complexity that is roughly proportional to the square of the number of generated cuts. It is thus natural to try to reduce the total number of columns by filtering identical columns or by eliminating or aggregating irrelevant elements.

Filter cut

The first order oracle may return similar cuts during the process. Since redundant cuts have a direct influence on the computation of the new query point, there is no reason to eliminate them. The idea of the filter cut approach consists in keeping only one copy of identical cuts and augmenting, in (6.14), the weight w on the logarithmic barrier associated to this cut. Thus, the weight w_i associated to a cutting plane $\Gamma_i - (A^T u - E^T z)_i$ is equal to the number of identical cuts returned by the oracle. Note that this approach has no influence on the analytic center. It reduces the amount of required memory and the complexity in the computation of the analytic centers.

Let us now discuss the way to detect identical cuts. One possibility is to compare, component by component, each new cut with the pool of generated columns in the previous iterations. When the number of columns is large, this

comparison requires costly computation. Instead we propose to use a hash table to assign a *signature* to each cut. Since identical cuts get a same value signature, it makes the detection of redundant columns undemanding.

Column elimination

The accumulation of cuts has a strong negative effect on the computational effort involved in interior point iterations. It seems that cuts that are far away from the current query points are likely to be good candidates for elimination.

The selection criterion for the elimination is based on that idea that constraints that are "far away" are not likely to be useful in the search for a solution, nor to play a significant role in the definition of the analytic center.

The distances between the analytic center and the cutting planes are given by the variables s. If s_i is much larger than the average of s, then column i contributes little to the solution (dually, the distance s_i between the analytic center and the cut is high). Such column is a good candidate for elimination. To make the elimination test robust, we use the median of s and eliminate the columns whose coefficient s_i is less than $1/\kappa$ times the median. In practice, we choose $\kappa = 4$. We also perform the test once every τ iterations. A good value, is $\tau = 20$.

Column aggregation

An alternative strategy to the elimination, consists in aggregating the less interesting cuts into one surrogate cut. The aggregation weights are the inverse of the current slacks. At each inner iteration, one can check whether one of the aggregated cut becomes active. If so, the culprit is expelled from the aggregated bundle and the search for the analytic center is pursued.

The selection criterion to the aggregation is the one discussed in the previous part (Column elimination). The idea of the column aggregation is to represent the unimportant constraints, say J_1, by a single constraint to reduce computation. The cuts in J_1 are not eliminated.

Assume (u^c, z^c) is the analytic center of the current localization set, and let $s^c = \Gamma - A^T u^c + E^T z^c$ be the associated slack variables. For the sake of simplicity, we assume that the analytic center is exact, so that $x^c s^c = w$. Let $J_1 \cup J_2 = \{1, \ldots, n\}$ be a partition of the set of constraints between important and unimportant constraints.

The aggregation process consists in taking a positive linear combination of the constraints in J_1. The weights are derived from the distance of the cuts

to the current analytic center. The aggregate cut is

$$A_r^T u - E_r^T z \leq \Gamma_r,$$

with

$$\begin{aligned} A_r &= A_{J_1} x_{J_1}^c = A_{J_1} W_{J_1} (s_{J_1}^c)^{-1}, \\ E_r &= E_{J_1} W_{J_1} (s_{J_1}^c)^{-1}, \\ \Gamma_r &= \Gamma_{J_1}^T (w_{J_1} (s_{J_1}^c)^{-1}). \end{aligned}$$

6.4.2 Implementation of active set strategies

We now investigate the possibility of fixing variables to given values between two outer iterations of ACCPM. This module is useful in the implementation of the active set strategy introduced in the previous chapter. Let us show now the influence of such module in the definition of the localization set and in the computation of the analytic center.

Let $J_1 \cup J_2 = \{1, \ldots, m\}$ be a partition of the set of variables. Then the cutting planes that support the function f_1 can be written as

$$A_{J_1}^T u_{J_1} + A_{J_2}^T u_{J_2} - E^T z \leq \Gamma. \tag{6.30}$$

Assuming that the optimal values of variables $u_{J_1} \in U_2$ are known and fixed. Then the analytic center is computed in the new localization set,

$$\mathcal{L}_k = \left\{ (u, z, \zeta) \mid A_{J_2}^T u_{J_2} - E^T z \leq \Gamma_{J_2},\ f_2(u) \leq \zeta,\ \pi^T z + \zeta \leq \bar{\theta},\ u_{J_2} \in U_2 \right\},$$

where

$$\Gamma_{J_2} = \Gamma - A_{J_1}^T u_{J_1}.$$

Thus the analytic center method defines the next query point to be the u component of the solution (u, z, ζ) to the minimization problem

$$\min \quad \frac{\rho}{2} ||u_{J_2} - \bar{u}_{J_2}||^2 - w_0 \log s_0 - \sum_{i=1}^{n} w_i \log s_i - \log \sigma + H(u_{J_2})$$

$$s_0 = \bar{\theta} - (\pi^T z + \zeta), \tag{6.31a}$$

$$s = \Gamma_{J_2} - A_{J_2}^T u_{J_2} + E^T z, \tag{6.31b}$$

$$\sigma = \zeta - f_2(u). \tag{6.31c}$$

Note that to make this module functional, the first order oracle has to get back to ACCPM the indexes and the values of fixed variables. In the pure cutting plane scheme, the oracle has not this practicality.

6.4.3 Fuzzy oracles

Following our convention, we consider, in this subsection, the case of a convex function $f(u)$ to be minimized without smooth component. We also assume that the first order oracle is in an aggregate mode.

It is sometimes the case that the first order oracle for $f(u)$ is imprecise. At a query point u^k, the oracle produces ϵ-subgradient defining a cutting plane such as

$$f(u) \geq f_k + a^T(u - u^k), \quad \forall u \in U, \tag{6.32}$$

where

$$f_k = f(u^k) - \epsilon.$$

This situation is displayed on Figure 6.2. Such a oracle usually results of a description of a function which is the maximum of a known collection of linear forms. Taking the maximum may be NP hard. Then, the optimization process may not have been carried to its end and the computed objective value may be a strict lower bound for the true value $f(u)$.

The vector a is not a subgradient, but the set

$$\left\{(u, z) \mid a^T u + (f_k - a^T u^k) \leq z\right\},$$

contains the epigraph of f; it is thus a valid cutting plane in a polyhedral description of f. To emphasize the fact that the cut is not connected to a correct value, we simply note it as

$$a^T u - z \leq \gamma,$$

with $\gamma = a^T u^k - f_k$. This cut is *loose*, by opposition with a *tight* cut, that would pass through the point $(u^k, f(u^k))$.

The information delivered by a loose cut is less accurate that the one delivered by a tight one, but it is still a valid one. This cut can be incorporated into the definition of the localization (6.6) without contradiction. The problem with a loose oracle lies in the definition of a valid upper bound $\bar{\theta}$, a notion that is essential in the definition of the localization set for ACCPM. Indeed, this set is also limited by a so-called hat cut, which bounds from above the epigraph variable by the best upper bound on f. This upper bound is computed from the function values returned by the oracle. If the oracle returns loose cuts, the function values may be under-estimated. The likely consequence: the localization doesn't contain the optimum of f as presented in Figure 6.3. The localization set may become empty that bring ACCPM to a halt.

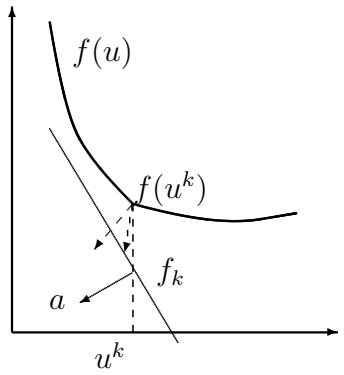

Figure 6.2: Under-estimate of f.

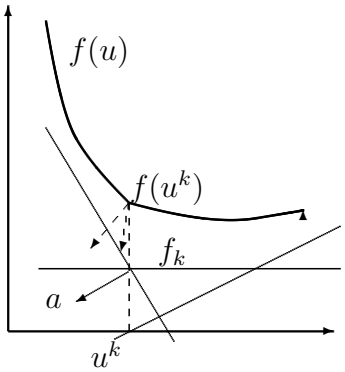

Figure 6.3: Localization set of f.

To analyze the possible contradictions, we assume that the first order oracle has been queried k times at $u^1, \ldots, u^k$. The oracle has returned corresponding sequence $(a^1, f_1), \ldots, (a^k, f_k)$, with the property that $f_i \leq f(u^i)$, $i = 1, \ldots, k$. Let $(a^1, \gamma_1), \ldots, (a^k, \gamma_k)$ be the sequence of cutting planes defining the current localization set

$$\mathcal{L}_k = \left\{(u, z) \mid (a^i)^T u - z \leq \gamma_i,\ i = 1 \ldots k\ , z \leq \bar{\theta}\right\}, \tag{6.33}$$

where

$$\bar{\theta} = \min_{i=1\ldots k} f_i.$$

Since the values f_i are not necessarily achieved by the true function f, we may well observe that some cut (a^j, γ_j) excludes a point (u^i, f_i):

$$(a^j)^T u^i - \gamma_j > f_i.$$

Such an event would reveal that the $f(u^i)$ was underestimated by the oracle, we have $f_i < f(u^i)$. Clearly the objective value delivered by the oracle has to be re-assessed. This can be done, either by changing the f_i, $i = 1, \ldots k$, or the right-hand sides γ_i in the localization set 6.33. To explain the patching procedure, we consider that $(\bar{u}, \bar{f})$ is the current proximal reference point. The second argument $\bar{f}$ designates what is currently considered as the best value for the upper bound. The upper bound $\bar{f}$ has to be updated by some technique we are going to discuss. Note that in this technique, if $\bar{u} = u^i$ for some i, we may well have $\bar{f} > f_i$, but $\bar{f} \leq f(u^i)$. This last property is very important, since it would prevent ACCPM from over-estimating the values of f.

Assume that (a^{k+1}, γ_{k+1}), with $\gamma_{k+1} = (a^{k+1})^T u^{k+1} - f_{k+1}$ is the cut introduced at u^{k+1}. The following cases may occur:

- the previously generated cuts give that $f(u^{k+1}) > f_{k+1}$;
- the new cut gives evidence that $f(\bar{u}) > \bar{f}$.

Let us examine the two cases successively.

In the first case, there is some cut (a^i, γ_i) such that

$$(a^i)^T u^{k+1} - \gamma_i > f^{k+1}.$$

This case is pictured below.

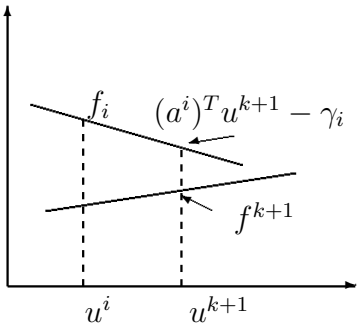

Figure 6.4: A former cut contradicts the recently computed value.

In that case, it may be tempting to shift the new cut upward, but we do not know if the shifted cut will still be a separating hyperplane for the epigraph. We certainly have $(a^i)^T u^{k+1} - \gamma_i \leq f(u^{k+1})$, but a^{k+1} may not

define a separating hyperplane. Therefore, the new cut is left at its initial position, but $(a^i)^T u^{k+1} - \gamma_i$ may be used to update the proximal reference point. Indeed, if

$$(a^i)^T u^{k+1} - \gamma_i < \bar{f},$$

then we may set

$$\bar{f} \leftarrow (a^i)^T u^{k+1} - \gamma_i \quad \text{and} \quad \bar{u} \leftarrow u^{k+1}.$$

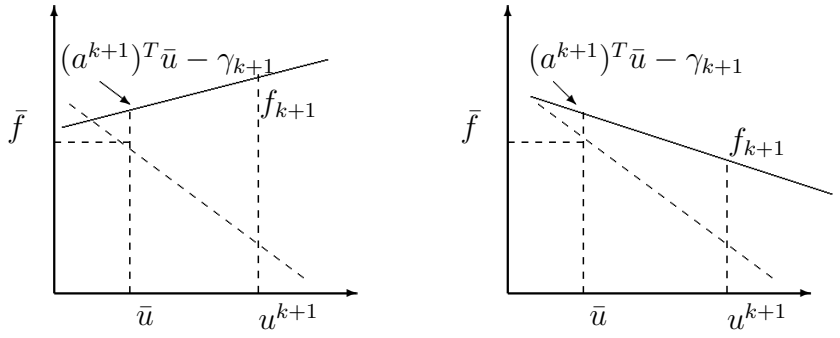

Figure 6.5: The new cut contradicts the function value at the proximal reference point.

The second case is subdivided into the two sub-cases pictured on Figure 6.5. In the first of the two sub-cases, the following inequalities hold

$$\bar{f} < (a^{k+1})^T\bar{u} - \gamma_{k+1} < f_{k+1}.$$

The proximal reference point remains $\bar{u}$ but the upper bound $\bar{f}$ is shifted upward to $(a^{k+1})^T\bar{u} - \gamma_{k+1}$. In the second of the two sub-cases, the following inequalities hold

$$f_{k+1} \leq \bar{f} < (a^{k+1})^T\bar{u} - \gamma_{k+1}.$$

The point u^{k+1} is the new candidate for the proximal reference point and f_{k+1} is the upper bound value.

The above procedure ensures that the localization set never gets empty. Note that the remedy we proposed has the effect of raising function values and thus the recorded upper bound. This may prevent convergence if the oracle keeps improving the function evaluation but never gets to the convergence precision. Indeed, one cannot hope for a relative duality gap smaller than the oracle precision. The user must take this into consideration in assessing a target convergence level.

6.5 Implementation details

6.5.1 ACCPM structure

Since the oracles are entirely user-defined, we do not include it in the description. The code has two main blocks: the first one computes query points; the second one organizes the dialog between the oracle and the query point generator. The code also includes an important initialization block. Let us describe these modules.

Initialization This module initializes the instance and the various parameters.

Query point generator This modules includes two submodules: the first one creates the localization set based on the information sent by the cut manager; the second one computes approximate proximal analytic centers.

Manager This module keeps track of the cuts generated by the first order oracle and of the current primal and dual coordinates of the analytic center. It also controls the parameters that are dynamically adjusted and computes criteria values that can be used by the user to stop the algorithm. Finally, it acts as a filter between the oracle and the query point generator.

6.5.2 Parameter settings in ACCPM

Few parameters of ACCPM are often critical in the applications: the weight w_0 on the logarithmic barrier associated to the epigraph cut, the proximal reference point $\bar{u}$ and the coefficient ρ of the proximal term.

Weight on hat cut

The localization set is bounded above by the special constraint $z + \zeta \leq \bar{\theta}$ in (6.6). It is easily checked that the hat cut makes a negative angle with the cutting planes. When the number of cutting planes increases, their total weight dominates the weight of the hat cut in (6.14). Thus, the floor cut tends to become active at the analytic center, with the possible effect of slowing the global convergence. To counteract this negative effect, the general strategy is to assign to w_0 a value equal to the number of generated cuts [59]. If we aim to a low precision, say 10^{-3}, we found that setting the weight to 50 times the total number of optimality cuts is more efficient.

Proximal center

The initial proximal center is the first query point. Thereafter, the proximal center is updated to the current query point whenever the oracle returns an objective function value that improves upon the best upper bound.

Coefficient of the proximal term

The management of the coefficient of the proximal term is more problem dependent. This point will be commented in the experimentations.

6.5.3 Implementation code

The code is written in Matlab; it has around 700 lines of code in the query point generator and 400 in the manager. Matlab is particularly efficient in dealing with linear algebra. Not much gain can be expected by translating the code into C++. However, a C version would make it easier to link ACCPM with oracles written in C or FORTRAN or to do an embedding of ACCPM within a larger optimization scheme (e.g., a branch and bound scheme). The code is the result of a continuing development efforts by teams at Logilab partly supported by Swiss NSF.

Part III

Numerical experiments

Chapter 7

The linear multicommodity flow problem

Contents

In this chapter, we apply ACCPM and the active set strategy to solve a collection of linear multicommotidy flow problems, in short LMCF, that can be found in the open literature. We consider the two classes of LMCF introduced in Chapter 2.

The first class of LMCF copes with a single kind of commodity and it has been intensively studied in this thesis. The numerical results presented here are reported from [13]. We used four categories of instances. The first two categories, `planar` and `grid`, gather artificial problems that mimic telecommunication networks. Some of them are very large. The third category is made of four small to medium size telecommunication problems. The last category includes six realistic traffic network problems; some of them are huge, with up to 13,000 nodes, 39,000 arcs and over 2,000,000 commodities.

For the second class of LMCF, which deals with physically different commodities, we report the first numerical results obtained with one category of problems, `Mnetgen`. Some Mnetgen instances are large network problems but the number of commodities is usually small.

The goals of the experiments are to test ACCPM, its enhancements and the active set strategy presented in Chapter 5. First, we test the impact of the proximal term in the ACCPM implementation. In the second set of experiments, we analyze the impact of column elimination while in the third one, we focus on the active set strategy. In the forth experiment, we combine column elimination and active set strategy to achieve the fastest computing time. Finally, in the last part, we benchmark our solution method with other methods used in the literature.

7.1 Models and relaxations

In this section, we present the two classes of LMCF used in the experiments. We also recall the dual problem resulting from the Lagrangian relaxation of the coupling constraints.

In the first class, the arc cost is given by a linear function of total arc flows and each commodity must be shipped from a single origin to a single destination. The commodity flows are constrained by mutual arc capacities but not by individual ones. The formulation of this problem is given by

$$\begin{aligned} \text{LMCF1}: \quad \min \quad & g(y) && \text{(7.1a)}\\ & y = \sum_{\kappa \in \mathcal{K}} x^\kappa, && \text{(7.1b)}\\ & N x^\kappa = d^\kappa, \quad \forall \kappa \in \mathcal{K}, && \text{(7.1c)}\\ & x^\kappa \geq 0, \quad \forall \kappa \in \mathcal{K}, && \text{(7.1d)} \end{aligned}$$

where the cost on arc a is given by

$$g_a(y_a) = \begin{cases} t_a y_a, & \text{if } y_a \in [0, c_a],\\ +\infty, & \text{otherwise.} \end{cases}$$

Note that, formulation (7.1) is equivalent to the linear traffic assignment flow problem introduced in Chapter 3.

In the second category of linear MCF problems, the commodities compete for mutual and individual arc capacities. Each commodity has multiple supply

nodes and demand nodes, and the arc cost is a linear function of each commodity flow. This LMCF is formulated as follows:

$$\begin{aligned} \text{LMCF2}: \quad \min \quad & g(y) + r^T x && (7.2a)\\ & y = \sum_{\kappa \in \mathcal{K}} x^\kappa, && (7.2b)\\ & N x^\kappa = d^\kappa, \quad \forall \kappa \in \mathcal{K}, && (7.2c)\\ & 0 \le x^\kappa \le c^\kappa, \quad \forall \kappa \in \mathcal{K}, && (7.2d) \end{aligned}$$

where the cost on arc a is given by

$$g_a(y_a) = \begin{cases} 0, & \text{if } y_a \in [0, c_a],\\ +\infty, & \text{otherwise.} \end{cases}$$

In view of Chapter 5, the Lagrangian relaxation of the coupling constraints in both formulations, i.e., (7.1b) and (7.2b), yields the general maximization problem

$$\max\{f_1(u) + f_2(u) \mid u \ge g'_+(0)\}, \tag{7.3}$$

where u is the vector of dual variables. ACCPM is then applied to solve (7.3) and get a primal feasible solution. The subproblem f_1 is a nonsmooth minimization problem revealed by a first order oracle and f_2 is a linear function accessed through a second order oracle. We will give the explicit formulations of f_1 and f_2 in the next section though the definition of the oracles.

7.2 Implementation issues

In this section, we review the main items in the implementation of our solution method. We defined the oracles in both formulations, and the strategy updating the proximal term in ACCPM.

7.2.1 First order oracle

The first order oracle solves the minimization problem f_1 at a query point u and get back to ACCPM the value of the objective function and a antisubgradient of $f_1(u)$. Let us distinguish the two classes of LMCF.

First order oracle for LMCF1

The subproblem f_1 resulting from the Lagrangian relaxation of (7.1) is given by

$$f_1(u) = \min_{x>0}\{\langle M^T u, x\rangle \mid Nx^\kappa = d^\kappa, \quad \forall \kappa \in \mathcal{K}\}. \tag{7.4}$$

Problem (7.4) appears to be a sum of simple shortest path problems, one per commodity. Then the first order oracle consists of $|\mathcal{K}|$ shortest path computations, using Dijkstra's algorithm [42]. This algorithm computes shortest paths from a single node to all other nodes in a directed graph. To compute the shortest paths for all commodities, we partition the commodities according to the origin node of the demand. This defines a subset of nodes $\mathcal{S} \subset \mathcal{N}$. We apply $|\mathcal{S}|$ times Dijkstra's algorithm, once for each $s \in \mathcal{S}$. For large graphs, most of the computational time is devoted to data handling. To speed-up computation, the algorithm is implemented with binary heap structures. This implementation is efficient enough, but probably not compare with the state-of-the-art. A better implementation would most likely improve the performance of the overall algorithm, but the focus of the paper is on the cutting plane method and not on shortest path computation.

Letting

$$x(u) = \arg\min_{x>0}\{\langle M^T u, x\rangle \mid Nx^\kappa = d^\kappa, \quad \forall \kappa \in \mathcal{K}\},$$

the first order oracle returns the objective value $f_1(u)$ and the antisubgradient $Mx(u)$.

First order oracle for LMCF2

The Lagrangian dual problem of (7.2) yields the following subproblem f_1,

$$f_1(u) = \min_{x>0}\{\langle M^T u + r, x\rangle \mid Nx^\kappa = d^\kappa,\ \forall \kappa \in \mathcal{K},\ x^\kappa \le c^\kappa,\ \forall \kappa \in \mathcal{K}\}. \tag{7.5}$$

In this case, f_1 is a sum of transshipment flow problems. At each iteration the first order oracle solves $|\mathcal{K}|$ transshipment problems, one per commodity, using the network optimizer of CPLEX 8.1. Since we solve the same transshipment problems at each iteration with different costs on the arcs, we can start from the previous basis. Starting with the advanced basis reduce significantly the number of CPLEX iterations and help performance.

Letting

$$x(u) = \arg\min_{x>0}\{\langle M^T u + r, x\rangle \mid Nx^\kappa = d^\kappa,\ \forall \kappa \in \mathcal{K},\ x^\kappa \le c^\kappa,\ \forall \kappa \in \mathcal{K}\},$$

the first order oracle returns the objective value $f_1(u)$ and the antisubgradient $Mx(u)$.

7.2.2 Second order oracle

When queried to a given point u, the second order oracle returns the function value $f_2(u)$ and its first derivative. In that situation the second derivative is null. In LMCF1, we objective value is given by $f_2(u) = c(t - u)$ while, in LMCF2, we have $f_2(u) = -cu$. In both cases, the first derivative is the constant $-c$.

7.2.3 Settings of the proximal coefficient in ACCPM

We used a fixed value for ρ, e.g. $\rho = 10^{-2}$. For problems with high tolerance requirement, say $\epsilon = 10^{-5}$, we need not update this value. Indeed, when approaching the optimal solution, (6.7) keeps generating exact upper bounds. If a lower precision is required, say 10^{-3}, it may happen that (6.7) only generates approximate upper bounds. Instead of iterating with the same ρ until (6.7) delivers an exact upper bound with the required precision, we find it convenient to use the approximate upper bound to signal closeness to the solution. We then switch to $\rho = 10^{-10}$. By lowering the impact of the proximal term, it makes it easier for ACCPM to find a primal feasible solution, i.e., an exact upper bound.

7.3 Test problems

In this section, we give the test problems used in the experiments for LMCF1 and LMCF2.

7.3.1 Test instances for LMCF1

We used four sets of test problems of traffic assignment to experiment our solution method on (7.1).

The first set, the `planar` problems, contains 10 instances and has been generated by Di Yuan to simulate telecommunication problems. Nodes are randomly chosen as points in the plane, and arcs link neighbor nodes in such a way that the resulting graph is planar. Commodities are pairs of origin and

destination nodes, chosen at random. Arc costs are Euclidean distances, while demands and capacities are uniformly distributed in given intervals.

The second set, the `grid` problems, contains 15 networks that have a grid structure such that each node has four incoming and four outgoing arcs. Note that the number of paths between two nodes in a grid network is usually large. The arc costs, commodities, and demands are generated in a way similar to that of `planar` networks. These two sets of problems are solved in [84]. The data can be downloaded from `http://www.di.unipi.it/di/groups/optimize/Data/MMCF.html`.

The third collection of problems is composed of telecommunication problems of various sizes. The cost functions for these problems are originally nonlinear. To make them linear, we use different techniques depending on the type of nonlinear cost function that was used. In the small `ndo22` and `ndo148` problems, the cost functions have a vertical asymptote. We use this asymptote as a natural capacity bound. Problem `904` is based on a real telecommunication network. It has 904 arcs and 11130 commodities and was used in the survey paper [102].

The last collection of problems is composed of transportation problems. The problems `Sioux-Falls`, `Winnipeg`, `Barcelona` are solved in [84]; there the demands of `Winnipeg` and `Barcelona` are divided, as in [84], by 2.7 and 3 respectively, to make those problems feasible. The last three problems, `Chicago-sketch`, `Chicago-region` and `Philadelphia` can be downloaded from `http://www.bgu.ac.il/~bargera/tntp/`. The data include an increasing congestion function that is not adapted to our formulation. This function uses capacity and "free flow time". We use this free flow time as unit cost. To turn those problems into linear ones we use the following strategy. For each problem, we divide all the demands by a same coefficient. We increase this coefficient until the problem becomes feasible with respect to the capacity constraints. We end up using coefficients 2.5, 6 and 7 for problems `Chicago-sketch`, `Chicago-region` and `Philadelphia`, respectively.

Table 7.1 (page 121) displays data on the four sets of problems. For each problem instance, we give the number of nodes $|\mathcal{N}|$, the number of arcs $|\mathcal{A}|$, the number of commodities $|\mathcal{K}|$, the cost value z^* of an optimal solution to (7.1) with a relative optimality gap less than 10^{-5}. Some instances are huge (over 500 millions variables and nearly 200 millions constraints for problem `Philadelphia`). The last column of Table 7.1 displays the percentage of saturated arcs, denoted $\%\frac{|\mathcal{A}_1^*|}{|\mathcal{A}|}$, at the optimum. Note that the figures in the last column are low, in particular for the real-life transportation problems.

Problem ID	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{K}\rvert$	z^*	% $\frac{\lvert\mathcal{A}_1^*\rvert}{\lvert\mathcal{A}\rvert}$
`planar` problems					
`planar30`	30	150	92	4.43508×10^7	9.3
`planar50`	50	250	267	1.22200×10^8	11.6
`planar80`	80	440	543	1.82438×10^8	24.3
`planar100`	100	532	1085	2.31340×10^8	16.3
`planar150`	150	850	2239	5.48089×10^8	27.5
`planar300`	300	1680	3584	6.89982×10^8	7.4
`planar500`	500	2842	3525	4.81984×10^8	2.0
`planar800`	800	4388	12756	1.16737×10^8	3.0
`planar1000`	1000	5200	20026	3.44962×10^9	9.6
`planar2500`	2500	12990	81430	1.26624×10^{10}	14.7
`grid` problems					
`grid1`	25	80	50	8.27323×10^5	8.7
`grid2`	25	80	100	1.70538×10^6	25.0
`grid3`	100	360	50	1.52464×10^6	4.2
`grid4`	100	360	100	3.03170×10^6	8.3
`grid5`	225	840	100	5.04970×10^6	3.7
`grid6`	225	840	200	1.04007×10^7	13.5
`grid7`	400	1520	400	2.58641×10^7	7.0
`grid8`	625	2400	500	4.17113×10^7	11.8
`grid9`	625	2400	1000	8.26533×10^7	16.3
`grid10`	625	2400	2000	1.64111×10^8	16.3
`grid11`	625	2400	3000	3.29259×10^8	11.0
`grid12`	900	3480	6000	5.77189×10^8	6.2
`grid13`	900	3480	12000	1.15932×10^9	8.0
`grid14`	1225	4760	16000	1.80268×10^9	3.5
`grid15`	1225	4760	32000	3.59353×10^9	4.0
Telecommunication-like problems					
`ndo22`	14	22	23	1.88237×10^3	9.0
`ndo148`	58	148	122	1.39500×10^5	0
`904`	106	904	11130	1.37850×10^7	9.2
Transportation problems					
`Sioux-Falls`	24	76	528	3.20184×10^5	2.6
`Winnipeg`	1067	2975	4345	2.94065×10^7	2.0
`Barcelona`	1020	2522	7922	3.89400×10^7	0.4
`Chicago-sketch`	933	2950	93513	5.49053×10^6	1.0
`Chicago-region`	12982	39018	2297945	3.06541×10^6	0.6
`Philadelphia`	13389	40003	1151166	1.65428×10^7	0.4

Table 7.1: Test problems: optimal value with 10^{-5} optimality gap.

7.3.2 Test instances for LMCF2

The set of problems used in the experiments for LMCF2 is composed of 26 instances. This set has been generated by the Kennington's Mnetgen generator [7] and it can be retrieved from `http://www.di.unipi.it/di/groups/optimize/Data/MMCF.html`. The parameters used to generate these instances can be found in [48]. They are considered to be difficult problems. They are also standard instances to test linear programming solvers in multicommodity flows.

Problem ID	$\|\mathcal{N}\|$	$\|\mathcal{A}\|$	$\|\mathcal{K}\|$	$\|U\|$	$\frac{\|\mathcal{A}^*_1\|}{\|U\|}$	z^*
			`Mnetgen` problems			
$\mathtt{M}_{64-4}$	64	524	4	407	0.5	192400.1
$\mathtt{M}_{64-8}$	64	532	8	425	3.7	394051.1
$\mathtt{M}_{64-16}$	64	497	16	391	10.0	1071474.9
$\mathtt{M}_{64-32}$	64	509	32	404	16.3	2146944.1
$\mathtt{M}_{64-64}$	64	511	64	405	25.9	4623138.4
$\mathtt{M}_{128-4}$	128	997	4	799	2.4	919643.1
$\mathtt{M}_{128-8}$	128	1089	8	851	6.3	1924133.8
$\mathtt{M}_{128-16}$	128	1114	16	911	13.9	4145079.4
$\mathtt{M}_{128-32}$	128	1141	32	906	19.6	9785961.1
$\mathtt{M}_{128-64}$	128	1171	64	936	30.2	19269824.2
$\mathtt{M}_{128-128}$	128	1204	128	979	39.8	40143200.8
$\mathtt{M}_{256-4}$	256	2023	4	1633	6.1	5026132.3
$\mathtt{M}_{256-8}$	256	2165	8	1764	11.7	9919483.2
$\mathtt{M}_{256-16}$	256	2308	16	1842	20.0	20692883.7
$\mathtt{M}_{256-32}$	256	2314	32	1829	27.0	45671076.0
$\mathtt{M}_{256-64}$	256	2320	64	1842	34.0	92249381.1
$\mathtt{M}_{256-128}$	256	1258	128	1870	38.8	190137259.9
$\mathtt{M}_{256-256}$	256	2204	256	1802	49.6	397882591.3
$\mathtt{M}_{512-4}$	512	4077	4	3288	9.3	21324851.1
$\mathtt{M}_{512-8}$	512	4373	8	3496	15.8	46339269.9
$\mathtt{M}_{512-16}$	512	4620	16	3669	27.0	96992237.2
$\mathtt{M}_{512-32}$	512	4646	32	3713	35	192941834.8
$\mathtt{M}_{512-64}$	512	4768	64	3853	44.1	412943158.7
$\mathtt{M}_{512-128}$	512	4786	128	3882	52.1	828013599.8
$\mathtt{M}_{512-256}$	512	4810	256	3827	58.4	1649356264.1
$\mathtt{M}_{512-512}$	512	4786	512	3820	63.3	3487588114.0

Table 7.2: Test problems.

Table 7.2 (page 122) displays data on this set of problems. For each problem instance, we give the number of nodes $|\mathcal{N}|$, the number of arcs $|\mathcal{A}|$, the number of commodities $|\mathcal{K}|$, and the exact optimal solution values z^* to (7.2) computed with CPLEX 8.1. In this set of instances, a part of the arcs has no mutual capacity. Then Table 7.2 also displays the number of capacity

constraints, equivalently, the number of dual variables, denoted $|U|$, and the percentage of saturated arcs, denoted $\%\frac{|\mathcal{A}_1^*|}{|U|}$, at the optimum.

7.4 Numerical experiments

The goals of the numerical study are five-fold. First, we compare two configurations of ACCPM in order to test the impact of the proximal term. In the first one we use a proximal term while the second setting has no proximal term but uses artificial bounds on the dual variables u to ensure compactness of the localization set. In this comparison we do not use the active set strategy or column elimination. In the second set of experiments, we analyze the impact of column elimination while in the third one, we focus on the active set strategy. In the forth set of experiments, we combine column elimination and the active set strategy to achieve the fastest computing time. Finally, in the last experiments, we benchmark our solution method with the most efficient methods used in the literature.

For all results using ACCPM, the tables give the number of outer iterations, denoted *Outer*, the number of Newton's iterations, or inner iterations, denoted *Inner*, the computational time in seconds, *CPU*, and the percentage of CPU time, denoted *%Or*, spent to compute the subproblem, i.e., the shortest path problems or the transshipment problems. When the active set strategy is activated, the working space of ACCPM is reduced to the active arcs only. Thus, we also give the percentage of arcs in the active set, $\%\frac{|\mathcal{A}_1|}{|\mathcal{A}|}$, and the percentage of saturated arcs, $\%\frac{|\mathcal{A}_1^*|}{|\mathcal{A}|}$, at the end of the solution process.

The ACCPM code we use has been developed in Matlab at the Logilab laboratory, while the shortest path algorithm is written in C. The tests were performed on a PC (Pentium IV, 2.8 GHz, 2 Gb of RAM) under Linux operating system.

7.4.1 Impact of the proximal term

In this subsection, we experiment the impact of the proximal term without using the active set strategy or column elimination. We compare ACCPM using a proximal term and ACCPM with boxes on variables. In the later case, we just set the proximal parameter to zero and introduce instead upper bounds on the variables to enforce compactness in the initial phase. In our experiments, the default upper bounds are chosen to be quite large, say 10^6,

to be inactive at the optimum. The results are displayed in Table 7.3 (page 124) for LMCF1 and in Table 7.4 (page 125) for LMCF2.

	ACCPM with a proximal term				**ACCPM with boxes**			
Problem ID	Outer	Inner	CPU	%Or	Outer	Inner	CPU	%Or
planar30	59	176	0.7	21	59	189	0.8	18
planar50	109	266	1.9	20	106	286	2.1	18
planar80	281	617	20.5	9	277	645	19.8	9
planar100	263	593	20.4	9	265	638	19.4	10
planar150	688	1439	330.0	2	700	1544	339.6	2
planar300	374	909	122.2	2	384	988	129.6	2
planar500	229	744	88.7	21	231	799	88.6	21
planar800	415	1182	557.2	16	419	1270	553.9	17
planar1000	1303	2817	7846.7	12	1314	2995	7896.8	12
planar2500	-	-	-	-	-	-	-	-
grid1	35	114	0.3	26	36	120	0.4	17
grid2	73	222	0.8	30	77	251	1.0	26
grid3	65	239	1.2	21	66	261	1.5	18
grid4	99	319	2.4	21	97	326	2.5	19
grid5	121	414	7.3	21	120	420	7.6	20
grid6	315	770	45.1	11	313	815	45.3	11
grid7	308	827	80.0	16	317	901	87.0	15
grid8	686	1601	893.9	8	691	1686	812.0	8
grid9	942	2082	1793.8	6	942	2159	1798.1	6
grid10	946	2096	1885.1	6	947	2173	1842.6	6
grid11	648	1515	715.1	10	647	1565	702.3	11
grid12	509	1341	658.5	18	507	1397	644.8	18
grid13	673	1629	1226.8	12	679	1728	1214.4	12
grid14	462	1363	843.6	22	469	1450	845.4	22
grid15	520	1450	1055.1	20	522	1529	1045.6	20
ndo22	18	59	0.1	12	18	62	0.1	12
ndo148	17	82	0.2	20	17	83	0.3	18
904	269	640	33.2	12	271	671	34.7	12
Sioux-Falls	30	95	0.3	24	32	117	0.4	19
Winnipeg	224	592	81.2	18	258	942	120.8	14
Barcelona	157	421	35.9	23	156	457	37.8	22
Chicago-sketch	180	493	79.2	47	182	523	83.6	45
Chicago-region	-	-	-	-	-	-	-	-
Philadelphia	-	-	-	-	-	-	-	-

Table 7.3: Impact of the proximal term.

In Table 7.3 (page 124), all problems, but three, are solved with a relative gap of 10^{-5}. None of the two configurations can solve planar2500, Chicago-region and Philadelphia, partly because too many cuts in the localization set jammed the memory space. Table 7.3 shows that the results are quite similar with boxes on variables and with a proximal term. The number of basic steps and the computational times are more or less the same. For this class of problems,

the two methods appear to be equivalent.

	ACCPM with a proximal term				**ACCPM with boxes**			
Problem ID	Outer	Inner	CPU	%Or	Outer	Inner	CPU	%Or
M_{64-4}	52	126	0.8	5	61	178	1.4	4
M_{64-8}	80	195	1.7	6	94	260	2.7	6
M_{64-16}	137	322	4.4	8	176	432	7.9	7
M_{64-32}	225	483	12.5	7	279	634	21.4	7
M_{64-64}	214	486	12.8	17	482	1010	77.0	8
M_{128-4}	96	227	3.4	5	152	348	10.2	4
M_{128-8}	154	350	10.0	5	199	484	17.3	6
M_{128-16}	320	703	51.3	5	448	992	112.9	4
M_{128-32}	305	689	50.2	8	788	1647	466.5	4
M_{128-64}	289	659	51.0	18	1185	2486	1542.0	4
$M_{128-128}$	264	607	54.2	33	1241	2547	1822.1	6
M_{256-4}	267	582	58.2	3	339	788	102.3	3
M_{256-8}	424	936	182.4	3	673	1486	556.0	2
M_{256-16}	421	962	201.2	4	1155	2451	2359.4	2
M_{256-32}	403	922	190.2	8	-	-	-	-
M_{256-64}	378	845	176.8	15	-	-	-	-
$M_{256-128}$	345	774	172.7	26	-	-	-	-
$M_{256-256}$	325	749	192.1	41	-	-	-	-
M_{512-4}	593	1278	789.0	2	835	1903	1940.5	2
M_{512-8}	588	1279	838.4	2	1471	3110	8216.8	2
M_{512-16}	556	1232	786.9	4	-	-	-	-
M_{512-32}	528	1198	751.4	6	-	-	-	-
M_{512-64}	485	1099	714.3	11	-	-	-	-
$M_{512-128}$	434	983	620.3	20	-	-	-	-
$M_{512-256}$	404	917	649.6	35	-	-	-	-
$M_{512-512}$	375	889	813.7	53	-	-	-	-

Table 7.4: Impact of the proximal term

Table 7.4 (page 125) gives results for LMCF2. In that set of experiments, we stopped the process after 1500 outer iterations. We observe that ACCPM using a proximal term outperforms ACCPM with boxes. The proximal term makes ACCPM possible to solve all the instances with 10^{-5} relative optimality in less than 1500 iterations.

As the proximal term gives better results than boxes on LMCF2, we will use it in all further experiments. It is also easier to manipulate the unique parameter ρ than implementing individual strategies to move the upper bounds on the variables.

7.4.2 Impact of column elimination

In this subsection, ACCPM solves the sets of problems using column elimination. We report the results in Tables 7.5 and 7.6 (pages 126 and 127). Column *Nb cuts* displays the number of remaining cuts at the end of the process while the last column *CPU Ratio* gives the improvement ratio of the CPU time of ACCPM without using column elimination strategy (see Tables 7.3 and 7.4, pages 124 and 125), and with column elimination strategy.

Problem ID	Nb cuts	Outer	Inner	CPU	%Or	**CPU ratio**
planar30	40	63	142	0.7	18	**1.0**
planar50	68	104	224	1.8	27	**1.1**
planar80	145	274	593	14.5	19	**1.4**
planar100	105	338	705	14.2	23	**1.4**
planar150	314	814	1641	136.9	16	**2.4**
planar300	171	383	803	80.7	25	**1.5**
planar500	103	221	495	59.4	37	**1.5**
planar800	188	388	845	281.3	39	**2.0**
planar1000	645	1058	2148	2861.4	20	**2.7**
planar2500*	1628	2156	4349	47355.8	18	-
grid1	31	35	97	0.3	18	**1.0**
grid2	46	60	132	0.6	19	**1.3**
grid3	47	63	193	1.2	17	**1.0**
grid4	60	93	249	2.0	18	**1.2**
grid5	81	123	364	6.1	20	**1.2**
grid6	164	308	683	28.5	21	**1.6**
grid7	182	312	749	55.2	23	**1.4**
grid8	385	706	1526	503.8	16	**1.8**
grid9	538	959	2008	1039.6	15	**1.7**
grid10	532	969	2022	1043.8	16	**1.8**
grid11	350	663	1453	434.6	22	**1.6**
grid12	280	520	1258	436.2	30	**1.5**
grid13	362	687	1575	773.5	25	**1.6**
grid14	231	478	1296	539.7	39	**1.6**
grid15	275	537	1367	696.9	36	**1.5**
ndo22	18	18	59	0.1	12	**1.0**
ndo148	17	17	82	0.2	20	**1.0**
904	119	254	533	19.1	21	**1.7**
Sioux-Falls	24	30	93	0.3	23	**1.0**
Winnipeg	118	227	613	54.5	25	**1.5**
Barcelona	105	156	438	28.9	27	**1.2**
Chicago-sketch	108	178	491	55.1	41	**1.4**
Chicago-region	460	1376	3030	44117.3	64	-
Philadelphia	326	885	1859	22937.2	66	-

* Problem solved with a relative optimality gap of 10^{-4}.

Table 7.5: Impact of column elimination.

Problem ID	Nb cuts	Outer	Inner	CPU	%Or	**CPU ratio**
M_{64-4}	38	52	126	0.8	5	**1.0**
M_{64-8}	62	80	195	1.5	7	**1.1**
M_{64-16}	93	136	326	3.4	8	**1.3**
M_{64-32}	95	226	510	7.8	13	**1.6**
M_{64-64}	123	211	490	8.4	27	**1.5**
M_{128-4}	76	96	228	3.3	5	**1.0**
M_{128-8}	73	156	357	8.1	5	**1.2**
M_{128-16}	150	316	702	25.3	12	**2.0**
M_{128-32}	144	308	712	26.9	15	**1.9**
M_{128-64}	117	284	710	30.7	29	**1.7**
$M_{128-128}$	158	272	640	40.1	45	**1.4**
M_{256-4}	131	274	601	38.8	3	**1.5**
M_{256-8}	186	418	958	89.2	5	**2.0**
M_{256-16}	181	417	979	94.2	8	**2.1**
M_{256-32}	165	411	1039	104.4	14	**1.8**
M_{256-64}	134	387	948	101.5	25	**1.7**
$M_{256-128}$	147	349	830	108.5	41	**1.6**
$M_{256-256}$	185	322	761	134.5	57	**1.4**
M_{512-4}	248	581	1322	331.2	2	**2.4**
M_{512-8}	241	575	1283	323.2	4	**2.6**
M_{512-16}	195	548	1273	316.9	8	**2.5**
M_{512-32}	183	553	1334	342.1	13	**2.2**
M_{512-64}	196	536	1498	414.2	20	**1.7**
$M_{512-128}$	172	454	1089	344.8	38	**1.8**
$M_{512-256}$	147	406	971	377.0	59	**1.7**
$M_{512-512}$	165	371	894	544.3	75	**1.5**

Table 7.6: Impact of column elimination

In both tables, all problems are solved with a relative optimality gap of 10^{-5}, except `planar2500` in Table 7.5 that is solved with a 10^{-4} precision. We observe a speed-up on all problems, with an average value 1.5. Since the number of outer iterations is about the same, the speed-up is due to a reduction of the computation time in ACCPM. It is apparent in comparing the proportion of time spent in the oracle (column 5 in Tables 7.3 and 7.4 and column 6 in Tables 7.5 and 7.6).

7.4.3 Impact of active set strategy

In this subsection, ACCPM solves the linear multicommodity flow problems with the active set strategy. Tables 7.7 and 7.8 (pages 128 and 129) shows the results with a relative optimality gap of 10^{-5}. `planar2500` in table 7.7 is solved with a relative gap of 1.2×10^{-5}. In the last column of the tables, we give the improvement ratio of the CPU time of ACCPM without using active

set strategy (see Tables 7.3 and 7.4), and with the active set strategy (see Tables 7.7 and 7.8).

Problem ID	$\%\frac{\lvert\mathcal{A}_1\rvert}{\lvert\mathcal{A}\rvert}$	$\%\frac{\lvert\mathcal{A}_1^*\rvert}{\lvert\mathcal{A}\rvert}$	Outer	Inner	CPU	%Or	**CPU ratio**
planar30	12.7	9.3	46	154	0.5	24	**1.5**
planar50	13.2	11.6	99	258	1.1	32	**1.7**
planar80	25.7	24.3	279	656	7.4	24	**2.8**
planar100	17.5	16.3	260	626	5.9	31	**3.5**
planar150	29.0	27.5	716	1597	93.8	8	**3.5**
planar300	9.1	7.4	343	835	15.4	18	**7.9**
planar500	2.6	2.0	140	359	12.8	87	**6.9**
planar800	3.6	3.0	317	786	81.7	85	**6.8**
planar1000	10.4	9.6	1249	2860	1244.9	36	**6.3**
planar2500*	15.8	14.7	2643	7160	34022.2	21	-
grid1	12.5	8.7	24	83	0.2	25	**1.4**
grid2	32.5	25.0	61	202	0.7	30	**1.2**
grid3	5.0	4.2	37	104	0.3	46	**3.8**
grid4	10.3	8.3	79	213	1.0	39	**2.4**
grid5	6.0	3.7	90	256	1.9	60	**3.9**
grid6	20.6	13.5	294	731	13.3	35	**3.4**
grid7	9.3	7.0	264	704	19.1	58	**4.2**
grid8	13.2	11.8	623	1465	155.5	37	**5.1**
grid9	18.1	16.3	907	2158	413.6	25	**4.3**
grid10	18.0	16.3	919	2209	432.7	26	**4.4**
grid11	12.2	11.0	569	1391	140.3	46	**5.1**
grid12	7.4	6.2	394	979	121.4	74	**5.4**
grid13	9.6	8.0	558	1333	209.4	59	**5.9**
grid14	4.4	3.5	310	767	139.8	89	**6.0**
grid15	4.9	4.0	364	902	173.7	86	**6.1**
ndo22	9.0	9.0	11	49	0.1	11	**1.5**
ndo148	1.7	0.0	2	13	0.01	35	**14.6**
904	13.1	9.2	321	984	15.9	30	**2.1**
Sioux-Falls	7.9	2.6	13	56	0.1	33	**3.2**
Winnipeg	2.9	2.0	158	393	12.8	83	**6.3**
Barcelona	0.5	0.4	35	111	2.1	86	**17.0**
Chicago-sketch	1.2	1.0	60	195	13.0	95	**6.1**
Chicago-region	1.1	0.6	683	1961	14684.2	98	-
Philadelphia	0.6	0.4	193	529	3125.2	99	-

* Problem solved with a relative optimality gap of 1.2×10^{-5}.

Table 7.7: Impact of active set strategy.

Tables 7.7 and 7.8 (pages 128 and 129) show that the active set strategy reduces the CPU time with a factor around 1.5 to 8 on most problems. The reduction is achieved in the computation of the analytic center. Indeed, the complexity of an inner iteration is roughly proportional to the square of the dimension of the working space. As shown in the second column of Tables 7.7 and 7.8, the dimension of the working space —measured by $|\mathcal{A}_1|$ at the end of

Problem ID	$\%\frac{\lvert\mathcal{A}_1\rvert}{\lvert U\rvert}$	$\%\frac{\lvert\mathcal{A}_1^*\rvert}{\lvert U\rvert}$	Outer	Inner	CPU	%Or	**CPU ratio**
M_{64-4}	4	0.5	17	62	0.2	14	**4.0**
M_{64-8}	11	3.7	74	215	1.2	14	**1.4**
M_{64-16}	19	10.0	132	342	2.8	16	**1.6**
M_{64-32}	28	16.3	146	368	4.4	18	**2.8**
M_{64-64}	41	25.9	148	368	5.7	31	**2.2**
M_{128-4}	9	2.4	67	185	1.4	10	**2.4**
M_{128-8}	16	6.3	149	347	5.8	9	**1.7**
M_{128-16}	26	13.9	183	457	11.8	13	**4.3**
M_{128-32}	36	19.6	274	847	35.0	12	**1.4**
M_{128-64}	46	30.2	204	512	22.7	29	**2.2**
$M_{128-128}$	51	39.8	200	507	30.7	44	**1.8**
M_{256-4}	17	6.1	201	484	18.1	8	**3.2**
M_{256-8}	27	11.7	260	641	40.0	8	**4.6**
M_{256-16}	37	20.0	307	799	73.7	9	**2.7**
M_{256-32}	44	27.0	285	712	72.5	15	**2.6**
M_{256-64}	50	34.0	265	644	74.3	25	**2.4**
$M_{256-128}$	56	38.8	261	646	94.8	37	**1.8**
$M_{256-256}$	62	49.6	255	644	119.7	51	**1.6**
M_{512-4}	23	9.3	357	840	140.8	5	**5.6**
M_{512-8}	34	15.8	428	996	261.2	5	**3.2**
M_{512-16}	44	27.0	431	1023	326.8	7	**2.4**
M_{512-32}	52	35.0	44	1083	402.5	10	**1.9**
M_{512-64}	60	44.1	406	954	400.6	16	**1.8**
$M_{512-128}$	64	52.1	393	1021	488.3	24	**1.3**
$M_{512-256}$	71	58.4	366	860	512.8	43	**1.3**
$M_{512-512}$	76	63.3	342	844	696.4	61	**1.2**

Table 7.8: Impact of active set strategy

the iterations— is often a small fraction of the total number of arcs. It is also interesting to note that the active set strategy leads to a satisfactory estimate of the set of saturated arcs at the optimum. Tables 7.7 and 7.8 show that the percentage of arcs in the active set and the percentage of saturated arcs are very close. Last but not least, the active set strategy has a favorable, though nonintuitive, influence on the total number of outer iterations.

7.4.4 Impact of active set strategy with column elimination

In this set of experiments, we combine column elimination and the active set strategy. The results are displayed on Tables 7.9 and 7.10 (pages 130 and 131). Column *Nb cuts* displays the number of remaining cuts at the end of the process. The last two columns of each table display CPU ratios. The next to the last column gives the ratio between the CPU times in Table 7.7 and

Table 7.9 for LMCF1 and in Table 7.8 and 7.10 for LMCF2. The last column does a similar comparison between Tables 7.3 and 7.4 and Tables 7.9 and 7.10, respectively. As expected, column elimination is efficient, not only because it decreases the time spent in computing analytic center, but also because it often permits a reduction of the total number of outer iterations. This last observation is rather surprising.

Problem ID	Nb cuts	Outer	Inner	CPU	%Or	**CPU ratios**	
planar30	30	48	150	0.4	29	**1.1**	**1.7**
planar50	61	97	252	1.1	32	**1.0**	**1.8**
planar80	144	283	703	6.5	28	**1.1**	**3.1**
planar100	110	257	641	5.2	34	**1.1**	**3.9**
planar150	296	820	1893	64.5	13	**1.5**	**5.1**
planar300	199	325	721	9.7	27	**1.6**	**12.6**
planar500	76	118	258	10.5	90	**1.2**	**8.5**
planar800	168	252	545	60.7	91	**1.3**	**9.2**
planar1000	628	890	1904	572.6	55	**2.2**	**13.7**
planar2500	2089	3009	7546	29457.3	28	-	-
grid1	24	24	83	0.2	25	**1.0**	**1.4**
grid2	41	52	164	0.6	31	**1.2**	**1.5**
grid3	32	32	88	0.3	39	**1.0**	**3.7**
grid4	47	66	175	0.7	46	**1.4**	**3.3**
grid5	58	75	194	1.6	58	**1.2**	**4.5**
grid6	165	239	607	8.3	46	**1.6**	**5.4**
grid7	159	228	582	13.7	70	**1.4**	**5.8**
grid8	374	528	1248	97.5	50	**1.6**	**8.1**
grid9	518	720	1680	212.8	38	**1.9**	**8.4**
grid10	499	722	1654	215.6	41	**2.0**	**8.7**
grid11	329	458	1094	84.9	62	**1.7**	**8.4**
grid12	232	329	803	88.9	84	**1.4**	**7.4**
grid13	343	460	1086	136.8	74	**1.5**	**9.0**
grid14	171	252	623	107.0	94	**1.3**	**7.9**
grid15	206	294	708	131.7	91	**1.3**	**8.0**
ndo22	11	11	49	0.1	11	**1.0**	**1.5**
ndo148	2	2	13	0.01	35	**1.0**	**14.6**
904	171	311	819	12.2	38	**1.3**	**2.7**
Sioux-Falls	2.6	13	56	0.1	33	**1.0**	**3.2**
Winnipeg	93	143	372	11.1	87	**1.2**	**7.3**
Barcelona	35	35	111	2.1	86	**1.0**	**17.0**
Chicago-sketch	44	65	170	12.9	96	**1.0**	**6.1**
Chicago-region	524	742	2080	15012.1	99	**1.0**	-
Philadelphia	127	192	525	3092.3	99	**1.0**	-

Table 7.9: Impact of active set strategy and column elimination.

Problem ID	Nb cuts	Outer	Inner	CPU	%Or	**CPU ratios**	
M_{64-4}	17	17	62	0.2	21	**1.0**	**4.0**
M_{64-8}	53	72	208	1.1	13	**1.1**	**1.5**
M_{64-16}	85	133	350	2.6	16	**1.1**	**1.7**
M_{64-32}	100	154	383	3.7	25	**1.2**	**3.4**
M_{64-64}	110	156	391	4.9	35	**1.2**	**2.6**
M_{128-4}	51	72	196	1.4	10	**1.0**	**2.4**
M_{128-8}	107	151	348	5.0	12	**1.2**	**2.0**
M_{128-16}	96	188	484	9.0	17	**1.3**	**5.7**
M_{128-32}	127	200	502	12.4	23	**2.8**	**4.0**
M_{128-64}	156	230	631	22.1	34	**1.0**	**2.3**
$M_{128-128}$	124	204	514	25.2	54	**1.2**	**2.2**
M_{256-4}	131	204	500	13.7	9	**1.3**	**4.2**
M_{256-8}	155	258	640	26.2	11	**1.5**	**7.0**
M_{256-16}	169	334	973	50.5	14	**1.5**	**4.0**
M_{256-32}	144	308	815	45.1	25	**1.6**	**4.2**
M_{256-64}	166	277	682	54.9	36	**1.3**	**3.2**
$M_{256-128}$	144	263	674	68.9	51	**1.4**	**2.5**
$M_{256-256}$	127	254	674	93.9	6	**1.3**	**2.0**
M_{512-4}	209	374	865	88.6	7	**1.6**	**8.9**
M_{512-8}	191	424	1025	121.0	9	**2.2**	**6.9**
M_{512-16}	280	453	1112	188.4	1	**1.7**	**4.2**
M_{512-32}	200	440	1124	183.2	20	**2.2**	**4.1**
M_{512-64}	199	401	975	198.3	30	**2.0**	**3.6**
$M_{512-128}$	217	399	985	263.6	43	**1.8**	**2.4**
$M_{512-256}$	178	365	906	340.6	64	**1.5**	**1.9**
$M_{512-512}$	146	321	882	487.4	78	**1.4**	**1.7**

Table 7.10: Impact of active set strategy and column elimination

7.4.5 Comparisons with other methods for LMCF1

In this subsection, we compare ACCPM with methods used in [84]. In [84], Larsson and Yuan propose an augmented Lagrangian algorithm (ALA) to generate feasible solutions with a reasonable precision. They compare their algorithm with a bundle method to find a "near optimal solution". They also provide a comparative study between four codes to generate solution with high precision. Their compare CPLEX 5.0, a specialized partitioning code (PPRN), the Dantzig-Wolfe decomposition and a bundle method. We refer the reader to Chapter 4 for a short description of these methods.

Since implementing, fine-tuning and testing these methods need a well knowledge of each method, we found more appropriate and more consistent to use the results of [84]. Then, we directly report their original computational times in the following tables to Benchmark ACCPM.

Since the machines are different (a Pentium IV, 2.8 GHz with 2 Gb of

RAM and a Sun ULTRASparc with 200 MHz processor and 2 GB of physical RAM), we used an artifact to estimate the speed ratio. We solved a large set of problems on the Pentium IV and on a Sun ULTRASparc with 500 MHz processor, the only Sun ULTRASparc we have at our disposal. We found a ratio of 4 between the Pentium IV and the Sun ULTRASparc 500, and we propose a 2.5 ratio between the two SUN's. Finally, we retain a factor of 10 between our machine and the one used in [84].

ACCPM vs. augmented Lagrangian relaxation

In this set of experiments, we compare ACCPM to ALA. In [84], the authors introduce the concept of "near optimal solution" to designate feasible solutions with a relative optimality gap around 10^{-3} (actually, ranging from 6.10^{-4} to 6.10^{-3}). Their solution method consists in running twice their algorithm, a first pass to compute a lower bound and a second pass to compute an upper bound. The total CPU time that is reported in Table 7.11 (page 133), is the sum of the two CPU times. To make a valid comparison, we aimed to results with a similar precision. Since the precision is moderate, we had to resort to the strategy defined in Section 7.2.

Table 7.11 (page 133) displays the comparative results on the instances used in [84]. The last column gives a CPU ratio between the CPU times of ACCPM and the CPU times of ALA reported in [84]. This CPU ratio includes the speed ratio between the two computers. Of course, those ratios are just indicative.

Table 7.11 shows that ALA is more efficient on the smaller instances[1] while the reverse holds for the larger ones[2]. In view of the active set strategy discussed in the subsection (7.4.3), we propose the following explanation for the behavior of ACCPM on small problems. Note that the percentage of time spent in the oracle vs. the master program steadily increases with the problem dimension. This suggests a possible computing overhead in ACCPM. Indeed, ACCPM is written in Matlab, while the oracle is implemented in C. Moreover, ACCPM is a general purpose code that is designed to handle a very large variety of problems. Consequently, the code contains a lot of structures that are costly to manipulate. However, on the large instances, the linear algebra operations dominate and Matlab is very efficient in performing them. An implementation of ACCPM in C would presumably improve the performance, essentially on the smaller instances.

[1] Problems smaller than `planar150` and `grid9` and also `Sioux-Falls`.

[2] Problems larger than `planar150` and `grid9` and `Winnipeg` and `Barcelona`.

		ACCPM					ALA		Ratio
Problem ID	$\%\frac{\lvert\mathcal{A}_1\rvert}{\lvert\mathcal{A}\rvert}$	Outer	Inner	CPU	%Or	Gap	CPU	Gap	**CPU**
planar30	11.3	35	178	0.4	21	0.0014	0.18	0.002	0.04
planar50	12.8	52	304	0.9	21	0.0023	1.28	0.0032	0.15
planar80	25.9	101	487	2.6	25	0.0045	12.25	0.0019	0.47
planar100	17.8	82	396	1.8	32	0.0026	12.51	0.0021	0.69
planar150	30	187	925	13.4	14	0.0044	61.80	0.0026	0.46
planar300	8.8	56	280	1.5	31	0.0018	103.09	0.0016	**7**
planar500	2.3	27	116	3.5	87	0.0010	211.22	0.0017	**8.5**
planar800	3.5	41	184	9.8	92	0.0014	1572.33	0.0017	**16**
planar1000	11.1	108	450	45.2	85	0.0021	3097.22	0.0018	**6.8**
planar2500	15.6	229	896	707.0	87	0.0018	34123.14	0.0013	**4.8**
grid1	11.1	17	104	0.17	20	0.0007	0.040	0.0062	0.02
grid2	28.7	26	161	0.42	21	0.0015	0.12	0.0060	0.03
grid3	4.4	12	63	0.17	29	0.0010	0.44	0.0022	0.26
grid4	9.2	23	106	0.35	34	0.0012	0.23	0.0029	0.07
grid5	6	20	115	3.8	7	0.0009	1.31	0.0012	0.03
grid6	15.6	35	190	1.1	51	0.0016	2.28	0.0023	0.2
grid7	11.9	23	151	1.4	70	0.0014	8.10	0.0016	0.6
grid8	17.2	42	219	4.9	78	0.0017	19.94	0.0020	0.4
grid9	19.5	50	261	7.4	76	0.0016	52.17	0.0023	0.7
grid10	22.9	48	233	7.3	80	0.0017	104.41	0.0022	**1.4**
grid11	16.7	34	189	4.7	82	0.0015	262.54	0.0014	**5.6**
grid12	13	22	140	5.6	90	0.0014	248.57	0.0019	**4.4**
grid13	15.6	26	157	6.5	89	0.0014	948.26	0.0019	**14.7**
grid14	7.8	16	102	6.7	95	0.0012	1284.79	0.0016	**19**
grid15	8.5	21	136	9.0	95	0.0012	2835.03	0.0014	**31.2**
Sioux-Falls	7.9	8	33	0.1	23	0.0023	0.47	0.0043	0.5
Winnipeg	3.4	65	203	5.2	84	0.00055	239.20	0.00061	**4.6**
Barcelona	1.6	30	85	1.9	82	0.00034	283.64	0.00057	**15**

Table 7.11: ACCPM vs. ALA.

Table 7.12 (page 134) displays the results for the other instances that are not considered in [84]. We solve them with a precision of 10^{-3}. For the two larger problems (Chicago-region and Philadelphia) we also give the results to obtain the first primal feasible solution without any condition on the relative gap. The reader will observe that our solution method produces a feasible solution for the larger problems in a very short time and with a reasonable relative optimality gap (around 10^{-2}). The computing time to gain one digit of accuracy is important. Yet, the overall time with a 10^{-3} relative precision is moderate.

Problem ID	$\%\frac{\lvert\mathcal{A}_1\rvert}{\lvert\mathcal{A}\rvert}$	Outer	Inner	CPU	%Or	Gap
ndo22	9	7	22	0.06	12	10^{-3}
ndo148	-	2	13	0.01	40	10^{-9}
904	14.5	248	721	10.6	35	10^{-3}
Chicago-sketch	1.2	12	50	2.6	96	10^{-3}
Chicago-region	2.9	12	274	213.8	99	0.026
Chicago-region	2.3	111	463	2239.8	99	10^{-3}
Philadelphia	1.2	8	145	112.5	99	0.012
Philadelphia	1.1	40	203	619.4	99	10^{-3}

Table 7.12: ACCPM.

ACCPM vs. bundle method

In this set of experiments, we make a comparison between ACCPM and the bundle method of [49] to compute a "near optimal solution" (as described in the previous experiment). Details of the bundle implementation are presented in [49]. In Table 7.13 (page 135), we report from [84] the results with the bundle method to reach the lower bound computed with ALA previously. For the sake of easier comparison, we report also from Table 7.11 the CPU times of ALA and ACCPM. The next to the last column gives a CPU ratio between the CPU times of ACCPM and the CPU times the bundle method. The last one is the CPU ratio between ACCPM and ALA reported from Table 7.11. These CPU ratios include the speed ratio between the two computers. Of course, those ratios are just indicative.

Table 7.13 shows that the bundle method is more efficient than ALA and ACCPM on smaller instances while the reverse holds on larger ones. Moreover, we note that the bundle method can not compute near optimal solution for larger instances. We do the same conclusion as in the previous set of experiments.

ACCPM vs. other methods

In [84], Larsson and Yuan compare four codes to compute solution with high precision. They use the solver CPLEX 5.0, the specialized primal partitioning method PPRN originates from [29], a Dantzig-Wole decomposition implemented in disaggregate form, and the bundle method of [49]. We refer the reader to [84] for more details.

We give in Table 7.14 (page 136) the original results reported in [84]. We use a "-" to denote that a problem could not be solved. The problems are

Problem ID	**Bundle**	**ALA**	**ACCPM**	**CPU ratios**	
planar30	0.16	0.18	0.4	0.04	0.04
planar50	0.85	1.28	0.9	0.09	0.15
planar80	22.82	12.25	2.6	0.9	0.47
planar100	99.97	12.51	1.8	**5.6**	0.69
planar150	392.59	61.80	13.4	**2.9**	0.46
planar300	321.39	103.09	1.5	**21.4**	7
planar500	699.12	211.22	3.5	**20.0**	8.5
planar800	-	1572.33	9.8	-	16
planar1000	-	3097.22	45.2	-	6.8
grid1	0.01	0.040	0.17	0.01	0.02
grid2	0.03	0.12	0.42	0.01	0.03
grid3	0.23	0.44	0.17	0.14	0.26
grid4	0.17	0.23	0.35	0.05	0.07
grid5	1.00	1.31	3.8	0.03	0.03
grid6	2.79	2.28	1.1	0.2	0.2
grid7	11.23	8.10	1.4	0.8	0.6
grid8	33.10	19.94	4.9	0.7	0.4
grid9	131.27	52.17	7.4	**1.8**	0.7
grid10	649.46	104.41	7.3	**8.9**	1.4
grid11	1897.87	262.54	4.7	**40.4**	5.6
grid12	2448.13	248.57	5.6	**43.7**	4.4
grid13	-	948.26	6.5	-	14.7
grid14	-	1284.79	6.7	-	19
grid15	-	2835.03	9.0	-	31.2
Sioux-Falls	1.58	0.47	0.1	**1.6**	0.5
Winnipeg	224.47	239.20	5.2	**4.3**	4.6
Barcelona	495.96	283.64	1.9	**26.1**	15

Table 7.13: ACCPM vs. Bundle.

solved by the bundle method with 10^{-6} relative gap. In the last column, we give the best results of ACCPM to compute a solution with 10^{-5} relative optimality gap. The settings include column elimination and active set strategy. In [84], the authors point out that the CPU times should not be paid too much attention, because the codes are developed under different circumstances and are of generality. Each code can be fine-tuned to produce improved CPU times for a specific set of instances. We also recall that the speed ratio between their machine and our machine is around 10.

As we can see, the bundle method and the Dantzig-Wolfe decomposition outperform the solver CPLEX and the specialized primal partitioning code PPRN. The huge size of the problems seems disastrous for direct methods. We conclude that ACCPM is very competitive with these efficient methods.

Problem ID	**Cplex 5.0**	**PPRN**	**Dantzig-Wolfe**	**Bundle**	**ACCPM**
planar30	2.71	4.90	0.10	0.26	0.4
planar50	75.97	236.66	0.77	2.45	1.1
planar80	5933.37	5952.27	9.29	67.67	6.5
planar100	30339.58	35013.75	20.7	196.97	5.2
planar150	946217.32	519993.41	192.48	1670.38	64.5
planar300	-	-	539.51	3824.19	9.7
planar500	-	-	1914.29	12968.20	10.5
planar800	-	-	47626.18	-	60.7
planar1000	-	-	142001.47	-	572.6
grid1	0.23	0.86	0.03	0.03	0.2
grid2	2.17	5.69	0.10	0.26	0.6
grid3	8.81	23.74	0.23	0.59	0.3
grid4	22.54	69.49	0.22	1.06	0.7
grid5	258.18	589.75	1.31	5.77	1.6
grid6	1433.91	3335.61	3.28	15.63	8.3
grid7	34515.28	42092.32	18.12	141.26	13.7
grid8	439702.66	325189.66	233.66	707.61	97.5
grid9	-	-	919.15	2598.50	212.8
grid10	-	-	1838.89	6438.61	215.6
grid11	-	-	4080.45	19207.20	84.9
grid12	-	-	6964.85	45339.80	88.9
grid13	-	-	33614.82	-	136.8
grid14	-	-	75640.48	-	107.0
grid15	-	-	281797.86	-	131.7
Sioux-Falls	74.45	179.40	0.40	2.77	0.1
Winnipeg	-	-	995.30	1828.82	11.1
Barcelona	-	-	79.85	523.86	2.1

Table 7.14: Computing times of some optimization code.

7.4.6 Comparisons with CPLEX for LMCF2

In that experiment, we compare ACCPM with CPLEX 8.1 which is considered in [26] as the most efficient method to solve the Mnetgen instances. In [26], Castro compares his interior point method (IPM) with CPLEX 6.5 on the Mnetgen instances. He observes than CPLEX outperforms IPM which is specialized for multicommodity flow problems. Here we use CPLEX 8.1 with dual solver and network options. We report in Table 7.15 (page 137) the CPU times obtained with CPLEX 8.1 and ACCPM using active set strategy and column elimination.

Table 7.15 shows that CPLEX is more efficient when the number of commodity is small. As the number of commodity is correlated with the number of saturated arcs at the optimum in this set of instances, we conclude that ACCPM is more efficient and competitive for problems that seems difficult.

Problem ID	**Cplex 8.1**	**ACCPM**	**CPU ratio**
M_{64-4}	0.02	0.2	0.1
M_{64-8}	0.04	1.1	0.02
M_{64-16}	0.17	2.6	0.07
M_{64-32}	0.72	3.7	0.2
M_{64-64}	5.42	4.9	**1.1**
M_{128-4}	0.06	1.4	0.04
M_{128-8}	0.16	5.0	0.03
M_{128-16}	0.8	9.0	0.09
M_{128-32}	8.32	12.4	0.7
M_{128-64}	42.13	22.1	**1.9**
$M_{128-128}$	89.26	25.2	**3.5**
M_{256-4}	0.16	13.7	0.01
M_{256-8}	0.60	26.2	0.02
M_{256-16}	4.41	50.5	0.09
M_{256-32}	20.08	45.1	0.4
M_{256-64}	66.50	54.9	**1.2**
$M_{256-128}$	89.85	68.9	**1.3**
$M_{256-256}$	131.43	93.9	**1.4**
M_{512-4}	0.43	88.6	0.005
M_{512-8}	1.72	121.0	0.01
M_{512-16}	7.04	188.4	0.04
M_{512-32}	54.22	183.2	0.3
M_{512-64}	78.54	198.3	0.4
$M_{512-128}$	140.06	263.6	0.5
$M_{512-256}$	180.95	340.6	0.5
$M_{512-512}$	413.10	487.4	0.8

Table 7.15: ACCPM vs. CPLEX 8.1

7.5 Conclusion

In the present chapter, we use ACCPM to solve two classes of linear multicommodity flow problems. The main new feature is the use of an active set strategy: it cuts down computational times by a factor from 2 to 14 and permits to solve the three larger instances that could not be solved previously. We also experiment two enhancements of ACCPM, the proximal term and the column elimination strategy, that make the method more efficient.

In the last experiments, we benchmark ACCPM with the most efficient methods used in the literature to solve the two classes of problems. First, we test the ability of our method to produce fast "near optimal solutions" to LMCF1 in the sense of [84]. In that paper, the authors compare an augmented Lagrangian algorithm (ALA) with a bundle method. We compared our results to theirs and observed an acceleration factor with ALA from 2 to 31 on the large problem instances. The factor is from 2 to 43 with the bundle method.

We also observe that price-directive methods (Dantzig-Wolfe, bundle method and ACCPM) outperform direct methods (CPLEX and Primal partitioning method) to produce high precision solution to LMCF1. ACCPM appears to be very competitive.

Finally, we compare ACCPM to CPLEX to compute solutions to LMCF2 with high precision. Since the experiments has been performed only on one set of instances, it seems difficult to generalized our observations. Other experiments on other sets of instances (PDS, Canad, JLF, Dimacs2pprn, BHV instances) would be required. Nevertheless, we observe that our method is competitive with CPLEX on larger problems.

Chapter 8

The nonlinear traffic assignment problem

Contents

In this chapter, we use ACCPM to solve nonlinear multicommodity flow problems, in short NLMCF. We carry the experiments with two objective functions: the Kleinrock congestion function used in telecommunications and the BPR delay function used in transportation.

The numerical results presented in this chapter are reported from [14]. We used four categories of instances. The first two categories, `planar` and `grid`, gather artificial problems that mimic telecommunication networks. Some of them are very large. The third category is made of four small to medium size telecommunication problems. The last category includes six realistic traffic network problems; some of them are huge, with up to 13,000 nodes, 39,000 arcs and over 2,000,000 commodities. Each set of instances is solved with both objective functions.

The goal of the experiments is to test ACCPM and its enhancements. The main feature is the implementation of a nonlinear cutting surface to handle the available second order information of the smooth objective function. We also perform the approximation scheme to apply the idea of the active set strategy on NLMCF. First, we test the impact of using a nonlinear cutting surface in the definition of the localization set. In the second set of experiments, we analyze the impact of the column elimination strategy. The third experiment focuses on the approximation scheme and the active set strategy. We observe that this approach is efficient only with the BPR function. Then we combine column elimination and active set strategy to achieve the fastest computing time with BPR. Finally, we benchmark our method with others used in the literature.

8.1 Models and relaxations

In the experiments, we consider the nonlinear traffic assignment problem introduced in Chapter 3. This problem is defined such as

$$\begin{aligned} \min \quad & g(y) && \text{(8.1a)}\\ & y = \sum_{\kappa \in \mathcal{K}} x^\kappa, && \text{(8.1b)}\\ & N x^\kappa = d^\kappa, \quad \forall \kappa \in \mathcal{K}, && \text{(8.1c)}\\ & x^\kappa \geq 0, \quad \forall \kappa \in \mathcal{K}, && \text{(8.1d)} \end{aligned}$$

where g is either the Kleinrock congestion function or the BPR delay function. Let us recall such functions. On the arc a, the Kleinrock function has the form

$$g_a(y_a) = \frac{y_a}{c_a - y_a},$$

and the BPR function is defined by

$$g_a(y_a) = r_a y_a \left(1 + \frac{\alpha}{\beta + 1} \left(\frac{y_a}{c_a}\right)^\beta\right).$$

The relaxation of constraints (8.1b) yields the Lagrangian dual problem

$$\max\{f_1(u) + f_2(u) \mid u \geq g'_+(0)\},$$

where u is the vector of dual variables. As shown in Chapters 5 and 6, the subproblem f_1 is a nonsmooth minimization problem revealed by a first order

oracle and f_2 is a smooth function accessed through a second order oracle. We will give the definition of f_1 and f_2 for the two considered objective functions in the next section through the description of the oracles.

8.2 Implementation issues

In this section, we review the main items in the implementation of our solution method. We defined the oracles in both formulations and the strategy updating the proximal term in ACCPM.

8.2.1 First order oracle

The first order oracle solves the minimization problem f_1 at a query point u and get back to ACCPM the value of the objective function and a antisubgradient of $f_1(u)$. The Kleinrock and the BPR functions lead to the same definition of f_1 given by

$$f_1(u) = \min_{x>0}\{\langle M^T u, x\rangle \mid Nx^\kappa = d^\kappa, \quad \forall \kappa \in \mathcal{K}\}. \tag{8.2}$$

The first order oracle consists of $|\mathcal{K}|$ shortest path computations, using Dijkstra's algorithm [42]. This algorithm computes shortest paths from a single node to all other nodes in a directed graph. To compute the shortest paths for all commodities, we partition the commodities according to the origin node of the demand. This defines a subset of nodes $\mathcal{S} \subset \mathcal{N}$. We apply $|\mathcal{S}|$ times Dijkstra's algorithm, once for each $s \in \mathcal{S}$. For large graphs, most of the computational time is devoted to data handling. To speed-up computation, the algorithm is implemented with binary heap structures. This implementation is efficient enough, but probably not compare with the state-of-the-art. A better implementation would most likely improve the performance of the overall algorithm, but the focus of the paper is on the cutting plane method and not on shortest path computation.

Letting

$$x(u) = \arg\min_{x>0}\{\langle M^T u, x\rangle \mid Nx^\kappa = d^\kappa, \quad \forall \kappa \in \mathcal{K}\},$$

the first order oracle returns the objective value $f_1(u)$ and the antisubgradient $Mx(u)$.

8.2.2 Second order oracle

At each iteration of the solution method, the second order oracle returns to ACCPM the function value and the first and second derivatives of f_2 at a given point point. The definition of f_2 is dependent of the primal objective function. See Section 5.4 (page 75) for a complete definition of f_2 associated to the Kleinrock congestion function and the BPR delay function.

8.2.3 Setting of the proximal coefficient in ACCPM

The initial value for the proximal coefficient ρ is 1. The rule to update this parameter is the following. When the method do not improve the upper bound $\bar{\theta}$ during few iterations, it may happen that the weight of the generated cuts is to large pushing the query point to far from the proximal reference point. To fix this behavior, we increase the impact of the proximal term multiplying ρ by 10 to ensure the new query point to remain closest from the best recorded value. It thus makes it easier for ACCPM to find a best dual solution, i.e., a best upper bound.

8.3 Test problems

We used four sets of test problems. The first set, the `planar` problems, contains 10 instances that have been generated by Di Yuan to simulate telecommunication problems. Nodes are randomly chosen as points in the plane, and arcs link neighbor nodes in such a way that the resulting graph is planar. Commodities are pairs of origin and destination nodes, chosen at random. Demands and capacities are uniformly distributed in given intervals.

The second set, the `grid` problems, contains 15 networks that have a grid structure such that each node has four incoming and four outgoing arcs. Note that the number of paths between two nodes in a grid network is usually large. Commodities, and demands are generated in a way similar to that of `planar` networks. These two sets of problems are used to solve the linear multicommodity flow problem in [13, 84]. The data include arc capacities and linear costs and can be downloaded from `http://www.di.unipi.it/di/groups/optimize/Data/MMCF.html`. We use directly these arc capacities in the definition of the Kleinrock function. To solve (8.1) with BPR function, we use the capacity as *practical capacity* and the linear cost as *free-flow travel time*. As suggested in [114], we use the parameter values $\alpha = 0.15$ and $\beta = 4$.

Problem ID	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{K}\rvert$	$z^*_{Kleinrock}$	z^*_{BPR}
`planar` problems					
`planar30`	30	150	92	40.5668	4.44549×10^7
`planar50`	50	250	267	109.478	1.21236×10^8
`planar80`	80	440	543	232.321	1.81906×10^8
`planar100`	100	532	1085	226.299	2.29114×10^8
`planar300`	300	1680	3584	329.120	6.90748×10^8
`planar500`	500	2842	3525	196.394	4.83309×10^9
`planar800`	800	4388	12756	354.008	1.16952×10^9
`planar1000`	1000	5200	20026	1250.92	3.41859×10^9
`planar2500`	2500	12990	81430	3289.05	1.23827×10^{10}
`grid` problems					
`grid1`	25	80	50	66.4002	8.33599×10^5
`grid2`	25	80	100	194.512	1.72689×10^6
`grid3`	100	360	50	84.5618	1.53241×10^6
`grid4`	100	360	100	171.331	3.05543×10^6
`grid5`	225	840	100	236.699	5.07921×10^6
`grid6`	225	840	200	652.877	1.05075×10^7
`grid7`	400	1520	400	776.566	2.60669×10^7
`grid8`	625	2400	500	1542.15	4.21240×10^7
`grid9`	625	2400	1000	2199.83	8.36394×10^7
`grid10`	625	2400	2000	2212.89	1.66084×10^8
`grid11`	625	2400	3000	1502.75	3.32475×10^8
`grid12`	900	3480	6000	1478.93	5.81488×10^8
`grid13`	900	3480	12000	1760.53	1.16933×10^9
`grid14`	1225	4760	16000	1414.39	1.81297×10^9
`grid15`	1225	4760	32000	1544.15	3.61568×10^9
Telecommunication-like problems					
`ndo22`	14	22	23	11.5631	1.87110×10^3
`ndo148`	58	148	122	151.926	1.40233×10^5
`904`	106	904	11130	33.4931	1.29197×10^7
Transportation problems					
`Sioux-Falls`	24	76	528	600.679	4.23133×10^6
`Winnipeg`	1067	2975	4345	1527.41	8.25673×10^5
`Barcelona`	1020	2522	7922	845.872	1.23277×10^6
`Chicago-sketch`	933	2950	93513	615.883	1.67484×10^7
`Chicago-region`	12982	39018	2297945	3290.55	2.58457×10^7
`Philadelphia`	13389	40003	1151166	2558.01	1.27810×10^8

Table 8.1: Test problems.

The third collection of problems is composed of telecommunication problems of various sizes. The small problems `ndo22` and `ndo148` are two practical problems solved in [52, 60]. Problem `904` is based on a real telecommunication network and was used in the survey paper [102]. This problem set is adapted to solve (8.1) with Kleinrock function. To solve (8.1) with BPR function, we use the capacity as *practical capacity* and also use it as *free-flow travel time*. We choose the parameter values $\alpha = 0.15$ and $\beta = 4$.

The last collection of problems is composed of six realistic transportation problems used in [13, 15, 38, 83]. Some of them are huge, with up to 13,000 nodes, 39,000 arcs and over 2,000,000 commodities. The data are adapted for the BPR function. They include *free-flow travel time*, *practical capacity* and the tuning parameters α and β. These problems, can be downloaded from `http://www.bgu.ac.il/~bargera/tntp/`. To solve (8.1) with Kleinrock function we use *practical capacity* as capacity and to turn these problems feasible with respect to the capacity, which is handled by the objective function, the demands are reduced as in [13, 83].

Table 8.1 (page 143) displays data on the four sets of problems. For each problem instance, we give the number of nodes $|\mathcal{N}|$, the number of arcs $|\mathcal{A}|$, the number of commodities $|\mathcal{K}|$, the optimal solution values to (8.1) $z^*_{Kleinrock}$ for the Kleinrock function and z^*_{BPR} for the BPR function, with a relative optimality gap less than 10^{-5}.

8.4 Numerical experiments

The main goal of our empirical study is to test the efficiency i) of using a nonlinear cutting surface, ii) of column elimination, iii) of active set strategy and iv) of column elimination and active set strategy jointly. We also use published results to benchmark the new algorithm.

We carry the experiments with the two congestion functions: the Kleinrock delay function and the BPR congestion function. For each solution strategy, we attempt to solve all problem instances contained in Table 8.1 with a 10^{-5} relative optimality gap. To benchmark the results with our best solution strategy, we use, for telecommunications problems (Kleinrock function), the results with the Projection Method reported in [102] and, for transportation problems (BPR function) several implementations of Frank-Wolfe algorithm reported in [38].

For all results using ACCPM, the tables give the number of outer iterations, denoted *Outer*, the number of Newton's iteration, or inner iterations, denoted

Inner, the computational time in seconds *CPU* and the percentage of CPU time, denoted *%Or*, spent to compute the shortest path problems. When the active set strategy is activated, the working space of ACCPM is reduced to the active arcs only. Thus, we give the percentage of arcs in the active set, $\%|\mathcal{A}_2|$, at the end of the solution process. We display also the error, denoted *Error*, leaded from the approximation with respect to the optimal solution of the original problem. Finally, when the elimination column is activated we display the number of remaining cuts, *Nb cuts*, at the end of the process.

The ACCPM code we use has been developed in Matlab, while the shortest path algorithm is written in C. The tests were performed on a PC (Pentium IV, 2.8 GHz, 2 Gb of RAM) under Linux operating system.

8.4.1 Impact of using a nonlinear cutting surface

In this subsection, we experiment the impact of the second order information in the solution method solving all the instances. We compare ACCPM using a second order oracle and ACCPM in which the smooth function is handled implicitly by the first order oracle as in the traditional approach.

The results are reported in Table 8.2 (page 146) for the Kleinrock function and in Table 8.3 (page 147) for the BPR function. In the two cases, we observe that the new approach outperforms the classical ACCPM. The larger problems are not solved by the classical ACCPM, partly because too many cuts in the localization set jammed the memory space.

8.4.2 Impact of column elimination

In this subsection, ACCPM solves the sets of problems using the column elimination. We report the results in Table 8.4 (page 148) for the Kleinrock function and in Table 8.5 (page 149) for the BPR function. The last column *CPU Ratio* displays the improvement ratio of the CPU time of ACCPM without column elimination (see Table 8.2 and 8.3), and with column elimination. We observe that column elimination speed-up the CPU time on all problems, with an average value 1.5. Since the number of outer iterations is about the same, the speed-up is due to a reduction of the computation time in ACCPM. It is apparent in comparing the proportion of time spent in the oracle.

	ACCPM with cutting surface				ACCPM without cutting surface			
Problem ID	Outer	Inner	CPU	%Or	Outer	Inner	CPU	%Or
planar30	93	197	1.1	22	832	1664	59.2	17
planar50	134	279	2.8	20	1234	2468	253.5	14
planar80	182	383	8.1	16	1965	3930	1381.3	11
planar100	187	392	10.2	17	2342	4684	2593.5	10
planar300	175	367	29.5	24	-	-	-	-
planar500	127	324	32.2	37	-	-	-	-
planar800	182	429	110.5	40	-	-	-	-
planar1000	381	869	568.1	26	-	-	-	-
planar2500	543	1224	3471.7	45	-	-	-	-
grid1	52	118	0.4	24	462	924	11.5	21
grid2	93	212	1.0	25	456	912	11.4	21
grid3	138	341	4.1	15	1713	3426	798.4	12
grid4	167	344	5.7	17	1613	3226	710.2	12
grid5	204	474	18.5	17	3409	6818	11005.7	8
grid6	333	686	55.9	14	3326	6652	10457.4	8
grid7	410	811	155.4	15	-	-	-	-
grid8	845	1783	1416.8	10	-	-	-	-
grid9	582	1269	576.9	15	-	-	-	-
grid10	432	964	300.6	20	-	-	-	-
grid11	261	581	106.4	29	-	-	-	-
grid12	201	409	106.7	41	-	-	-	-
grid13	222	454	128.7	39	-	-	-	-
grid14	204	414	173.2	48	-	-	-	-
grid15	203	414	172.8	48	-	-	-	-
ndo22	12	86	0.2	7	173	346	1.5	27
ndo148	70	361	1.3	7	737	1474	45.7	17
904	135	294	10.4	27	-	-	-	-
Sioux-falls	140	345	1.7	24	533	1410	13.0	15
Winnipeg	338	988	215.0	14	-	-	-	-
Barcelona	253	678	101.1	15	-	-	-	-
Chicago-sketch	145	370	48.6	41	-	-	-	-
Chicago-region	190	500	8621.9	94	-	-	-	-
Philadelphia	279	822	13094.4	89	-	-	-	-

Table 8.2: Impact of the cutting surface (Kleinrock delay function).

	ACCPM with cutting surface				**ACCPM without cutting surface**			
Problem ID	Outer	Inner	CPU	%Or	Outer	Inner	CPU	%Or
planar30	56	256	1.2	25	202	502	5.7	15
planar50	96	422	3.4	15	328	775	21.8	11
planar80	159	665	13.0	12	651	1400	165.0	7
planar100	101	423	6.5	16	629	1341	172.3	8
planar300	98	358	18.3	21	1101	2398	1980.2	6
planar500	42	164	10.0	37	986	2261	2548.9	7
planar800	88	299	51.1	43	2155	4513	32792.7	5
planar1000	192	552	209.6	37	-	-	-	-
planar2500	364	1744	3099.2	38	-	-	-	-
grid1	24	108	0.4	26	148	329	1.8	30
grid2	49	202	0.8	25	238	557	3.5	27
grid3	31	121	0.7	23	259	586	13.4	16
grid4	57	216	1.7	22	386	859	35.3	10
grid5	60	199	3.5	23	557	1230	174.2	7
grid6	125	385	11.5	16	918	1931	660.0	6
grid7	102	307	15.1	22	1111	2405	1872.9	5
grid8	158	422	49.5	23	1982	4119	13010.7	3
grid9	211	597	97.3	22	2379	4923	22202.1	3
grid10	207	586	94.8	24	2404	4965	22925.9	3
grid11	138	413	47.9	31	1966	4082	12873.5	4
grid12	107	323	52.7	46	2002	4369	20391.3	4
grid13	117	340	59.1	44	2300	4785	28038.6	4
grid14	84	274	61.9	56	1995	4179	26729.3	5
grid15	93	293	70.2	55	1588	4011	17449.6	4
ndo22	4	35	0.1	0	75	287	0.8	23
ndo148	6	45	0.2	0	171	390	3.6	24
904	93	316	8.4	19	802	1729	470.2	6
Sioux-falls	80	411	2.0	19	366	1057	10.1	20
Winnipeg	81	298	16.3	36	1307	2783	3352.8	8
Barcelona	56	245	10.4	29	925	2040	1493.7	8
Chicago-sketch	72	265	20.4	48	1828	4075	11891.0	5
Chicago-region	332	1502	10606.5	64	-	-	-	-
Philadelphia	287	1250	7469.9	63	-	-	-	-

Table 8.3: Impact of the cutting surface (BPR congestion function).

Problem ID	Nb cuts	Outer	Inner	CPU	%Or	**ratio**
planar30	54	114	293	1.1	26	**1.0**
planar50	67	164	385	2.2	27	**1.3**
planar80	126	239	544	6.5	24	**1.2**
planar100	95	208	481	6.0	24	**1.7**
planar300	104	205	484	22.2	32	**1.3**
planar500	75	129	319	24.1	47	**1.3**
planar800	99	179	447	77.1	54	**1.4**
planar1000	207	369	836	303.6	43	**1.9**
planar2500	339	567	1292	2398.5	64	**1.5**
grid1	45	59	194	0.5	27	0.8
grid2	55	108	242	0.8	46	**1.0**
grid3	71	121	307	2.3	22	**1.8**
grid4	71	188	394	3.1	31	**1.8**
grid5	137	203	472	12.0	21	**1.5**
grid6	113	389	828	24.3	26	**2.3**
grid7	244	471	1024	90.9	22	**1.7**
grid8	311	876	1845	384.0	21	**3.7**
grid9	344	646	1397	305.5	24	**1.9**
grid10	274	474	1039	199.8	29	**1.5**
grid11	228	270	599	96.6	32	**1.1**
grid12	189	212	431	107.2	43	**1.0**
grid13	207	234	478	125.8	41	**1.0**
grid14	183	212	430	166.5	51	**1.0**
grid15	181	205	418	161.5	52	**1.1**
ndo22	12	12	86	0.2	7	**1.0**
ndo148	47	73	176	0.8	25	**1.6**
904	85	138	300	7.6	27	**1.4**
Sioux-falls	73	144	353	1.6	25	**1.1**
Winnipeg	202	384	1155	135.3	19	**1.6**
Barcelona	182	261	732	71.2	19	**1.4**
Chicago-sketch	77	130	340	30.1	53	**1.6**
Chicago-region	92	194	597	8672.1	95	**1.0**
Philadelphia	109	290	826	13021.8	93	**1.0**

Table 8.4: Impact of column elimination (Kleinrock delay function).

Problem ID	Nb cuts	Outer	Inner	CPU	%Or	ratio
planar30	26	57	276	1.2	21	**1.0**
planar50	34	95	418	2.9	23	**1.2**
planar80	39	167	611	6.7	21	**1.9**
planar100	25	99	383	4.3	23	**1.5**
planar300	42	98	345	12.6	33	**1.5**
planar500	24	40	156	8.6	44	**1.2**
planar800	36	76	250	34.2	56	**1.5**
planar1000	72	177	519	135.1	51	**1.5**
planar2500	120	346	1350	1657.9	66	**1.9**
grid1	15	24	105	0.4	33	**1.0**
grid2	27	52	194	0.8	33	**1.0**
grid3	20	30	114	0.7	27	**1.0**
grid4	20	55	194	1.5	28	**1.1**
grid5	30	61	194	2.4	9	**1.5**
grid6	39	139	380	6.8	24	**1.7**
grid7	34	100	283	9.2	32	**1.6**
grid8	51	166	441	30.6	35	**1.6**
grid9	57	217	578	46.4	40	**2.1**
grid10	52	206	560	45.6	44	**2.1**
grid11	39	144	398	30.0	48	**1.6**
grid12	35	100	287	34.5	60	**1.5**
grid13	37	122	346	44.2	60	**1.3**
grid14	28	88	270	53.1	67	**1.2**
grid15	33	99	293	59.9	68	**1.2**
ndo22	4	4	35	0.1	0	**1.0**
ndo148	6	6	45	0.2	0	**1.0**
904	37	100	311	6.2	27	**1.4**
Sioux-falls	28	83	346	1.4	30	**1.4**
Winnipeg	36	76	259	10.6	47	**1.5**
Barcelona	20	55	231	7.9	38	**1.3**
Chicago-sketch	36	84	272	18.3	57	**1.1**
Chicago-region	124	342	1327	8224.8	85	**1.2**
Philadelphia	95	254	1031	4962.5	83	**1.5**

Table 8.5: Impact of column elimination (BPR congestion function).

8.4.3 Impact of the active set strategy

In this subsection, we experiment the combination of the approximation scheme and the active strategy to solve (8.1) with the Kleinrock and BPR functions. This strategy turns out to be efficient only with the BPR function, but not with the Kleinrock function. The very steep slope of Kleinrock close its asymptote leads to a larger spread of the flows on the arcs. All arcs turn out to be moderately congested and the compound function does not provide a satisfactory approximation. Table 8.6 (page 151) gives the computational results using the active set strategy on the approximate BPR function. The last column, shows the improvement ratio of CPU time of ACCPM without active set strategy (see Table 8.3), and with the active set strategy. The value of g^* in (5.26) is empirical. We get $g^* = 10^8$ for telecommunication instances (`planar`, `grid` and telecommunications-like problems) and $g^* = 10^7$ for traffic networks.

Table 8.6 (page 151) shows that active set strategy speed-up the CPU time on all problems (excepted the smaller ones) until 8.4. This speed-up is partly due to the large number of inactive arcs in the optimal solution. The number of dual variables handled by ACCPM is usually lower than 60%. A second explanation is the reduction of the total number of outer iteration around 10%. Removing the inactive arcs from the Lagrangian relaxation seems to make ACCPM easier the converge. The important point is that the quality of the optimal solution is not affected by the approximation, i.e., the computed optimal solution for the approximate problem is also a optimal solution with 10^{-5} optimality gap for the original problem. For three instances, the approximation doesn't ensure a 10^{-5} optimality gap but it is also traduced by a larger decrease of number of outer iterations, of size of the active set, and obviously of CPU time. This observation shows the difficulties to guaranty a given optimality gap in a static approximation scheme.

8.4.4 Impact of active set strategy with column elimination

In this set of experiments, we combine both column elimination and the active set strategy. Obviously, since active set strategy is not efficient with Kleinrock function, we solve only (8.1) with the BPR congestion function. The settings of the active set strategy and the column elimination are those used in the previous subsections. The results are displayed on Table 8.7 (page 152) . In the last column of the table, we give the improvement ratio of the CPU time

Problem ID	Error	$\%\lvert A_2\rvert$	Outer	Inner	CPU	%Or	**ratio**
planar30	$< 10^{-5}$	53	58	299	1.2	36	**1.0**
planar50	$< 10^{-5}$	63	99	473	2.9	33	**1.2**
planar80	$< 10^{-5}$	62	154	803	9.4	24	**1.4**
planar100	$< 10^{-5}$	60	97	536	4.9	29	**1.3**
planar300	$< 10^{-5}$	44	82	399	9.2	38	**2.0**
planar500	$< 10^{-5}$	26	38	216	5.8	62	**1.7**
planar800	$< 10^{-5}$	26	77	383	29.6	68	**1.7**
planar1000	$< 10^{-5}$	40	168	775	132.2	54	**1.6**
planar2500	$< 10^{-5}$	43	323	1996	2063.3	51	**1.5**
grid1	$< 10^{-5}$	86	25	118	0.4	32	**1.0**
grid2	$< 10^{-5}$	99	49	210	0.8	31	**1.0**
grid3	$< 10^{-5}$	40	25	122	0.6	33	**1.1**
grid4	$< 10^{-5}$	50	52	222	1.7	34	**1.0**
grid5	$< 10^{-5}$	47	50	224	2.3	37	**1.5**
grid6	$< 10^{-5}$	63	107	400	8.6	31	**1.3**
grid7	$< 10^{-5}$	52	76	312	8.8	39	**1.7**
grid8	$< 10^{-5}$	54	113	437	26.1	38	**1.9**
grid9	$< 10^{-5}$	64	178	612	60.8	34	**1.6**
grid10	$< 10^{-5}$	66	195	661	74.5	34	**1.3**
grid11	$< 10^{-5}$	61	139	475	41.9	43	**1.1**
grid12	$< 10^{-5}$	51	87	330	34.6	59	**1.5**
grid13	$< 10^{-5}$	57	113	448	53.4	51	**1.1**
grid14	$< 10^{-5}$	44	78	323	47.2	69	**1.3**
grid15	$< 10^{-5}$	49	88	346	56.9	66	**1.2**
ndo22	$< 10^{-5}$	50	5	46	0.1	-10	**1.0**
ndo148	$< 10^{-5}$	75	6	49	0.2	17	**1.0**
904	$< 10^{-5}$	32	93	358	5.1	42	**1.6**
Sioux-falls	$< 10^{-5}$	100	69	336	1.7	30	**1.2**
Winnipeg	1.3×10^{-5}	33	48	246	6.8	49	**2.4**
Barcelona	$< 10^{-5}$	24	37	241	3.8	51	**2.7**
Chicago-sketch	$< 10^{-5}$	52	65	262	16.9	56	**1.2**
Chicago-region	5×10^{-4}	34	55	698	1261.3	87	**8.4**
Philadelphia	5×10^{-5}	54	97	1234	2157.9	74	**3.5**

Table 8.6: Impact of the active set strategy (BPR congestion function).

of ACCPM without using the two options (see Table 8.3), and with using them. As expected, column elimination reduces the computational time. As in subsection 8.4.2, it decreases the time spent in computing the analytic centers.

Problem ID	Error	$\%\|A_2\|$	Nb cuts	Outer	Inner	CPU	%Or	**ratio**
planar30	$< 10^{-5}$	53	26	58	315	1.2	30	**1.0**
planar50	$< 10^{-5}$	63	26	92	475	2.9	31	**1.2**
planar80	$< 10^{-5}$	63	42	196	896	7.6	33	**1.7**
planar100	$< 10^{-5}$	60	36	98	506	4.2	32	**1.5**
planar300	$< 10^{-5}$	44	35	96	434	9.4	50	**1.9**
planar500	$< 10^{-5}$	26	16	34	196	5.1	70	**2.0**
planar800	$< 10^{-5}$	27	31	78	363	27.1	77	**1.9**
planar1000	$< 10^{-5}$	20	78	177	815	113.6	64	**1.9**
planar2500	$< 10^{-5}$	43	118	354	1898	1540.0	73	**2.0**
grid1	$< 10^{-5}$	86	16	25	117	0.4	34	**1.0**
grid2	$< 10^{-5}$	97	21	56	229	0.8	31	**1.0**
grid3	$< 10^{-5}$	40	18	26	125	0.7	33	**1.0**
grid4	$< 10^{-5}$	50	27	51	197	1.4	40	**1.2**
grid5	$< 10^{-5}$	47	22	45	207	1.9	39	**1.8**
grid6	$< 10^{-5}$	63	47	115	384	6.4	41	**1.8**
grid7	$< 10^{-5}$	52	37	73	290	6.7	47	**2.3**
grid8	$< 10^{-5}$	54	45	116	413	19.1	50	**2.6**
grid9	$< 10^{-5}$	64	45	177	680	39.0	48	**2.5**
grid10	$< 10^{-5}$	67	42	188	605	40.9	55	**2.3**
grid11	$< 10^{-5}$	61	44	139	455	28.4	59	**1.7**
grid12	$< 10^{-5}$	51	32	89	321	29.4	70	**1.8**
grid13	$< 10^{-5}$	57	35	112	418	38.9	64	**1.5**
grid14	$< 10^{-5}$	44	33	74	324	40.8	75	**1.5**
grid15	$< 10^{-5}$	49	35	84	328	47.4	75	**1.5**
ndo22	$< 10^{-5}$	50	5	5	46	0.1	-12	**1.0**
ndo148	$< 10^{-5}$	75	6	6	49	0.1	23	**2.0**
904	$< 10^{-5}$	32	37	114	358	3.5	55	**2.4**
Sioux-falls	$< 10^{-5}$	100	31	93	413	1.5	40	**1.3**
Winnipeg	1.4×10^{-5}	33	21	47	251	5.7	51	**2.9**
Barcelona	$< 10^{-5}$	24	18	35	213	3.1	54	**3.3**
Chicago-sketch	$< 10^{-5}$	52	29	68	269	15.1	64	**1.3**
Chicago-region	5×10^{-4}	34	22	55	681	1229.5	89	**8.6**
Philadelphia	5×10^{-5}	54	41	99	1185	2004.0	81	**3.7**

Table 8.7: Active set strategy and column elimination (BPR congestion function).

8.4.5 Comparisons with other methods

In this subsection, we compare ACCPM with a Projection Method on telecommunications problems using the Kleinrock delay function. We also compare ACCPM with several implementations of Frank-Wolfe algorithm on transportation problems with the BPR congestion function.

ACCPM vs. the Projection Method

In this experiment, we compare the results of our solution method ACCPM using column elimination with the results of the Projection Method (PM) reported in [102]. As in [102], we solve problem `904` with a varying load factor to generate different demands. Table 8.8 (page 153) gives the load factors we use and the corresponding optimal value with a 10^{-5} relative optimality gap.

Problem ID	Load factor	$z^*_{Kleinrock}$
`904`	1	33.4931
`904(1.5)`	1.5	52.2678
`904(2)`	2	72.6437
`904(2.5)`	2.5	94.8839
`904(3)`	3	119.305

Table 8.8: Test problems.

In [102], the authors compare a previous version of ACCPM implemented in [60] with the Flow Deviation Method [85], the Projection Method [22], and the Proximal Decomposition Method [92]. In this comparative study, the Projection Method (PM) appears to be the most efficient method to solve the `904` instances. We use the figures reported in [102] for PM and ACCPM [1].

The computational tests in [102] are performed on an IBM RISC/System 6000. We report the original computing times in Tables 8.9 (page 154) and 8.10 (page 154). In order to compare these results with those we obtain with the new version of ACCPM on a Pentium IV, we have performed benchmark computations according to BYTEmark[2]. We found a ratio 14. We use this

[1]The ACCPM version [60] works on a disaggregated form of the objective function. It exploits the sparsity in the master problem to cope with the very large number of generated cuts. In the disaggregated approach, the oracle generates as many cuts as the number of commodities at each outer iteration. In the case of problem `904`, this means 11130 cuts.

[2]BYTE Magazine's BYTEmark benchmark program (release 2) available at http://www.byte.com/bmark/bmark.htm.

value to compare the speeds of the algorithms in the two columns entitled *ratio* in Tables 8.9 and 8.10. These ratios are just indicative.

	ACCPM					previous ACCPM*		
Problem ID	Nb cuts	Outer	Inner	CPU	%Or	Outer	CPU	**ratio**
904	85	138	300	7.6	27	14	3233	**30.4**
904(1.5)	105	150	314	9.8	24	14	3441	**25.0**
904(2)	109	172	357	11.8	24	14	3186	**19.3**
904(2.5)	107	189	398	13.9	24	13	3276	**16.8**
904(3)	101	192	415	12.6	24	13	3544	**20.1**

* Tests performed in [102] on an IBM RISC/System 6000 machine.

Table 8.9: ACCPM and Previous ACCPM.

	ACCPM					PM*		
Problem ID	Nb cuts	Outer	Inner	CPU	%Or	Outer	CPU	**ratio**
904	85	138	300	7.6	27	579	380	**3.6**
904(1.5)	105	150	314	9.8	24	663	434	**3.2**
904(2)	109	172	357	11.8	24	688	471	**2.8**
904(2.5)	107	189	398	13.9	24	741	558	**2.9**
904(3)	101	192	415	12.6	24	691	501	**2.8**

* Tests performed in [102] on an IBM RISC/System 6000 machine.

Table 8.10: ACCPM and PM.

Tables 8.9 and 8.10 shows that the new ACCPM using column elimination outperforms the previous version of ACCPM and improves the computational time of the Projection Method with a ratio 3.

ACCPM vs. Frank-Wolfe algorithm

In this experiment, we compare ACCPM with the results obtained in [38] with different versions of Frank-Wolfe algorithm: a classical Frank-Wolfe method (FW), a conjugate direction Frank-Wolfe method (CFW) and a bi-conjugate Frank-Wolfe method (BFW). These methods outperform the Frank-Wolfe method implemented in [15]. We solve the same set of transportation problems as in [38], with BPR function with a 10^{-4} relative gap.

Since we have not at our disposal the machine used in [38], we cannot compare the computational times. To get an idea of performance, we focus on the number of iterations to solve the problems. In this experiment, we do

not use the active set strategy to have perfect control on the precision of the optimal solution.

Problem ID	ACCPM	BFW	CFW	FW
Sioux-falls	44	124	357	1869
Winnipeg	39	163	243	838
Barcelona	31	41	34	51
Chicago-sketch	27	21	17	24
Chicago-region	115	43	53	126

Table 8.11: ACCPM vs. Frank-Wolfe.

The results displayed on Table 8.11 (page 155) show that ACCPM is competitive with the implementations of Frank-Wolfe algorithm in term of number of iterations, except for the last instance. ACCPM is more efficient on the smaller instances.

8.5 Conclusion

In this chapter, we experimented two important modifications of the analytic center cutting plane method to solve nonlinear multicommodity flow problems: a cutting surface to handle the smooth component of the Lagrangian dual objective and an approximation scheme for the nonsmooth component of that objective. The approximation scheme is coupled with an active set strategy that leads to an expression of the Lagrangian dual in a space of smaller dimension. The new approach considerably improves the performance of the former implementation of ACCPM. It compares favorably with the known most efficient methods.

The present results suggests that possible further improvements could be achieved using the approximation/active set approach with a different linearization scheme for the cost function. Conceptually, this linearization could be performed around points that are dynamically chosen to lead more efficient approximations. This will be the object of further researches.

Chapter 9

Conclusion

Contents

9.1 Summary

The goal of the thesis is to solve large scale multicommodity flow problems (MCF). A new version of the analytic center cutting plane method (ACCPM) is used to solve the Lagrangian dual problem. In the present dissertation, we propose important features that considerably improve the performances of the solution method on linear and nonlinear MCF.

In Chapter 5, we introduce an active set strategy to reduce the dual working space. It exploits the following observation. In the linear case, many capacity constraints are not saturated at the optimum. This results in null dual variables at the optimum. The active set strategy proposes to estimate the set of null variables during the process. It cuts down the computational times by a factor from 2 to 14 and makes ACCPM possible to solve the largest instances of linear MCF.

This strategy is extended to nonlinear functions with a linear part near the origin, i.e, the *compound functions*. In Chapter 5, we implement a approximation scheme that replaces nonlinear functions by compound ones. It makes then

possible to use the active set strategy. The combination of the approximation scheme and the active set strategy turns out to be efficient with the BPR delay function but not with the Kleinrock congestion function. It speeds up the computational time until 8.4 with BPR functions.

The other important feature, presented in Chapter 6, is the implementation in ACCPM of a nonlinear cutting surface to exploit the second order information of the smooth component of the Lagrangian dual objective. This scheme is applied on the nonlinear MCF whose the Lagrangian dual is the sum of a nonlinear smooth function and a nonsmooth one. The new approach considerably improves the performance of the former implementation of ACCPM and makes it possible to solve all the instances.

In all the experiments, the combination of a proximal term and the column elimination strategy induces an improvement of the computational times. The proximal term is an alternative of introducing boxes on dual variables to ensure compactness of the localization set in ACCPM. The column elimination permits to reduce the number of cutting planes in ACCPM and makes the computation of the analytic centers less costly.

In Chapters 7 and 8, we compare our solution method with others used in literature. It compares favorably with the known most efficient methods as well on linear MCF as in nonlinear MCF. In the linear case, we compare ACCPM with direct approaches, i.e, CPLEX and primal partitioning method, and with price-directive methods, i.e, Dantzig-Wolfe decomposition and augmented Lagrangian and bundle methods. In the nonlinear case, we make a comparison with a Frank-Wolfe algorithm and with a Projection method that is a variant of the simplicial decomposition. We conclude that our solution method is very competitive with these methods.

9.2 Future directions

Let us review some directions for future research.

For the sake of completeness, we would like to study more intensively the class of linear MCF in which the commodities have multiple origins and multiple destinations. In Chapter 7, we experimented our solution method on only one set of these instances. In literature, there are many other sets of such problems, i.e, PDS, Canad, JLF, Dimacs2pprn and BHV instances. In general, these instances are more compact and direct methods seem more appropriate than price-directive approaches. We cannot expect too much progress on the class of problems.

The second extension of the thesis would be to study the traffic assignment with elastic demands. The thesis introduced the way to solve this problem and it would be interesting to observe the behavior of ACCPM. We may expect nice results.

Then, the present study suggests that possible further improvements could be achieved using the approximation/active set approach with a different linearization scheme for the cost function. Conceptually, this linearization could be performed around points that are dynamically chosen to lead more efficient approximations. We may expect that other approximations would be efficient with all objective functions and not only with the BPR function.

Finally, we believe that the method can be accelerated by exploiting the fact that the objective is the sum of independent components. In previous studies on the nonlinear multicommodity flow problem [60, 102], the implementation exploited the fact that the function f is the sum of $|\mathcal{K}|$ independent functions. It associates with each one of them an epigraph variable and optimality cuts. A much richer information is thus transferred to the master program that enables convergence in very few outer iterations (often less than 15 on large problems). In the meantime, the computation time of the analytic centers dramatically increases. As a result, 95% of the time is spent in the master program that computes analytic centers [60, 102]. In the present dissertation, the proportion is just reverse: we observe that 95% of the time is spent in solving shortest path problems for the larger problem instances. If we could achieve a better balance between the two components of the algorithm, we would improve performance.

Appendices

A Self-concordant analysis

In this chapter, we show that the objective function of (6.14) is a self-concordant function when f_2 is connected to our multicommodity flow context.

A.1 Properties of self-concordant function

Let us first recall the definition of a self concordant function.

Definition 7. *(Definition 4.1.1 of [98])*
f is a self-concordant function on $\mathcal{C}$ with the constant $M_f \geq 0$ if the inequality

$$|\langle f'''(x)[h]h, h\rangle| \leq M_f ||h||^3_{f''(x)}, \tag{A.1}$$

holds for any $x \in \mathcal{C}$ and $h \in \mathbb{R}^m$.

Let us mention several simple examples of self-concordant functions [98] which are relevant in our situation:

- The linear function $f(x) = ax + b,\ x \in \mathbb{R}^m$, is self-concordant on $\mathbb{R}^m$ with $M_f = 0$.
- The convex quadratic function $f(x) = b + \langle a, x\rangle + \frac{1}{2}\langle Ax, x\rangle,\ x \in \mathbb{R}^m$, where A is a diagonal semi positive definite matrix, is a self-concordant function on $\mathbb{R}^m$ with parameter $M_f = 0$.
- The logarithmic function $f(x) = -\ln x,\ x \in \mathbb{R}^+$, is a self-concordant function with $M_f = 2$.

We now introduce some elementary properties and a useful result on self-concordant functions.

Theorem 8. *(Theorem 4.1.1 of [98])*
Let f and g be self-concordant functions with parameters M_f and M_g. The function $\alpha f(x) + \beta g(x)$ is self-concordant on the intersection of the domains of f and g, with parameter $\max(\frac{M_f}{\sqrt{\alpha}}, \frac{M_g}{\sqrt{\beta}})$.

Theorem 9. *(Theorem 4.1.2 of [98])*
Let $\mathcal{A}(x) = Ax + b$ be a linear operator. Assume that $f(x)$ is a self-concordant function with parameters M_f. Then the function $\Phi(x) = f(\mathcal{A}(x))$ is also a self-concordant function with parameter M_f.

Lemma 1. *(Lemma 3.2 of [57])*
Let g be a convex function defined on $\mathcal{C}$ and suppose that there exists a constant γ such that

$$\langle g'''(x)[h]h, h\rangle \leq 3\gamma\langle g''(x)h, h\rangle\sqrt{\sum_{i=1}^{m}\frac{h_i^2}{x_i^2}}, \quad \forall x \in \mathcal{C} \text{ and } h \in \mathbb{R}^m. \tag{A.2}$$

We have that

$$f(x) = -\log(\zeta - g(x)) - \sum_{i=1}^{m}\log x_i, \tag{A.3}$$

is self-concordant on $\mathcal{C}$ with parameter

$$M_f = \max(1, \frac{\gamma + 1 + 1/\gamma}{\sqrt{3 + 4/\gamma + 2/\gamma^2}}). \tag{A.4}$$

A.2 Self-concordant results

Let us rewrite the objective function of (6.14),

$$G(u, z, \zeta) = \frac{\rho}{2}||u - \bar{u}||^2 - w_0\log s_0 - \sum_{i=1}^{n} w_i \log s_i - \log\sigma + H(u), \tag{A.5}$$

with

$$\begin{aligned} s_0 &= \bar{\theta} - (\pi^T z + \zeta), \\ s_i &= \Gamma_i - (A^T u - E^T z)_i, \quad i = \{1, \ldots, n\}, \\ \sigma &= \zeta - f_2(u). \end{aligned}$$

From Theorem 8 and Theorem 9, it is easy to show that

$$\frac{\rho}{2}||u - \bar{u}||^2 - w_0\log s_0 - \sum_{i=1}^{n} w_i \log s_i,$$

is self-concordant. Then self-concordance of G is directly related to the self-concordance of

$$-\log\sigma + H(u).$$

Recall that the above function is defined by the smooth objective function f_2 and its feasible set U_2, the global self-concordance of G is problem dependent. Thus, we propose to show that G is a self concordant function in the case of smooth functions f_2 connected to the multicommodity flow context. We focus on the three main cases:

- f_2 is a linear function.
- f_2 is the Frenchel conjugate of the Kleinrock function.
- f_2 is the Frenchel conjugate of the BPR function.

Theorem 10. *Consider the linear function* $f(x) = \sum_{i=1}^{m} c_i x_i$, *with* $c_i > 0$ *and* $x_i > 0$, *for all* $i = 1, \ldots, m$, *we have that the function*

$$g(x, \zeta) = -\log(\zeta - f(x)) - \sum_{i=1}^{m} \log(x_i),$$

is a self-concordant function with constant $M_f = 2$.

Proof. Easy proof. □

Theorem 11. *Let* $f(x) = \sum_{i=1}^{m}(1 + c_i x_i - 2\sqrt{c_i x_i})$, *with* $c_i > 0$ *and* $x_i \in [\frac{1}{c_i}, +\infty[$, *for all* $i = 1, \ldots, m$, *be the Frenchel conjugate of the Kleinrock function. Then the function*

$$g(x, \zeta) = -\log(\zeta - f(x)) - \sum_{i=1}^{m} \log(x_i - \frac{1}{c_i}),$$

is a self-concordant function.

Proof. Let us start with the negative conjugate of Kleinrock function.

$$\frac{3\gamma D^2 f(v)[h,h]\sqrt{\sum_{i=1}^{n} \frac{h_i^2}{v_i^2}}}{D^3 f(v)[h,h,h]} = \frac{-\frac{3}{2}\gamma \sum_{i=1}^{n} h_i^2 \sqrt{c_i}(v_i + \frac{1}{c_i})^{-\frac{3}{2}} \sqrt{\sum_{i=1}^{n} \frac{h_i^2}{v_i^2}}}{\frac{3}{4}\sum_{i=1}^{n} h_i^3 \sqrt{c_i}(v_i + \frac{1}{c_i})^{-\frac{5}{2}}},$$

from the standard inequality $\sum_{i=1}^{n} x_i^2 \geq \frac{1}{n}(\sum_{i=1}^{n} |x_i|)^2$, we have

$$\begin{aligned}
\frac{3\gamma D^2 f(v)[h,h]\sqrt{\sum_{i=1}^n \frac{h_i^2}{v_i^2}}}{D^3 f(v)[h,h,h]} &\geq \frac{-\frac{3}{2}\gamma \sum_{i=1}^n h_i^2 \sqrt{c_i}(v_i+\frac{1}{c_i})^{-\frac{3}{2}} \frac{1}{\sqrt{n}} \sum_{i=1}^n h_i v_i^{-1}}{\frac{3}{4}\sum_{i=1}^n h_i^3 \sqrt{c_i}(v_i+\frac{1}{c_i})^{-\frac{5}{2}}}, \\
&\geq \frac{-\frac{3}{2\sqrt{n}}\gamma \sum_{i=1}^n h_i^3 \sqrt{c_i}(v_i+\frac{1}{c_i})^{-\frac{3}{2}} v_i^{-1}}{\frac{3}{4}\sum_{i=1}^n h_i^3 \sqrt{c_i}(v_i+\frac{1}{c_i})^{-\frac{5}{2}}}, \\
&= \frac{-\frac{3}{2\sqrt{n}}\gamma \sum_{i=1}^n h_i^3 \sqrt{c_i}(v_i+\frac{1}{c_i})^{-\frac{5}{2}}(v_i+1) v_i^{-1}}{\frac{3}{4}\sum_{i=1}^n h_i^3 \sqrt{c_i}(v_i+\frac{1}{c_i})^{-\frac{5}{2}}}, \\
&\geq \frac{-\frac{3}{2\sqrt{n}}\gamma \sum_{i=1}^n h_i^3 \sqrt{c_i}(v_i+\frac{1}{c_i})^{-\frac{5}{2}}}{\frac{3}{4}\sum_{i=1}^n h_i^3 \sqrt{c_i}(v_i+\frac{1}{c_i})^{-\frac{5}{2}}}, \\
&\geq -\frac{2\gamma}{\sqrt{n}}.
\end{aligned}$$

From Lemma 1, we have that

$$g(x,\zeta) = -\log(\zeta - f(x)) - \sum_{i=1}^{m} \log(x_i - \frac{1}{c_i}),$$

is a self-concordant function with constant

$$M_f = \max(1, \frac{-\frac{\sqrt{n}}{2}}{\sqrt{3 - 2/(-\frac{\sqrt{n}}{2})}}).$$

□

Theorem 12. *Let* $f(x) = \sum_{i=1}^{m} (\frac{c_i(x_i - r_i)^{\frac{\beta+1}{\beta}}}{(\alpha r_i)^{\frac{1}{\beta}}} \frac{\beta}{\beta+1})$, *with* $c_i > 0$ *and* $x_i > 0$, *for all* $i = 1, \ldots, m$, *be the Frenchel conjugate of the BPR function. Then the function*

$$g(x,\zeta) = -\log(\zeta - f(x)) - \sum_{i=1}^{m} \log(x_i),$$

is a self-concordant function.

Proof. We now prove the same result for the negative conjugate of BPR function.

$$\frac{3\gamma D^2 f(v)[h,h]\sqrt{\sum_{i=1}^n \frac{h_i^2}{v_i^2}}}{D^3 f(v)[h,h,h]} = \frac{3\gamma \sum_{i=1}^n h_i^2 s_i v_i^{\frac{1-\beta}{\beta}} \sqrt{\sum_{i=1}^n \frac{h_i^2}{v_i^2}}}{\sum_{i=1}^n h_i^3 \frac{1-\beta}{\beta} s_i v_i^{(1-2\beta)/\beta}},$$

where $s_i = c_i/\beta(t_i\alpha)^{\frac{1}{\beta}},\ \forall i = 1 \dots n$. From the standard inequality $\sum_{i=1}^n x_i^2 \geq \frac{1}{n}(\sum_{i=1}^n |x_i|)^2$, we have

$$\begin{aligned}
\frac{3\gamma D^2 f(v)[h,h]\sqrt{\sum_{i=1}^n \frac{h_i^2}{v_i^2}}}{D^3 f(v)[h,h,h]} &\geq \frac{3\gamma \sum_{i=1}^n h_i^2 s_i v_i^{\frac{1-\beta}{\beta}} \frac{1}{\sqrt{n}} \sum_{i=1}^n h_i v_i^{-1}}{\frac{1-\beta}{\beta} \sum_{i=1}^n h_i^3 s_i v_i^{(1-2\beta)/\beta}}, \\
&\geq \frac{\frac{3}{\sqrt{n}}\gamma \sum_{i=1}^n h_i^3 s_i v_i^{(1-2\beta)/\beta}}{\frac{1-\beta}{\beta} \sum_{i=1}^n h_i^3 s_i v_i^{(1-2\beta)/\beta}}, \\
&\geq \frac{3(1-\beta)\gamma}{\beta\sqrt{n}}.
\end{aligned}$$

From Lemma 1, we have that

$$g(x,\zeta) = -\log(\zeta - f(x)) - \sum_{i=1}^m \log(x_i),$$

is a self-concordant function with constant

$$M_f = \max(1, \frac{\frac{\beta\sqrt{n}}{3(1-\beta)}}{\sqrt{3 - 2/\frac{\beta\sqrt{n}}{3(1-\beta)}}}).$$

□

Then the objective function (6.14a) is self-concordant in our multicommodity flow context. This property can be exploited to solve the minimization problem (6.14).

Bibliography

[1] H.Z. Aashtiani. The multi-modal traffic assignment problem. *Ph.D. dissertation, Operation Research Center, Massachusetts Institute of Technology, Cambridge, MA*, 1979.

[2] H.Z. Aashtiani and T.L. Magnanti. Equilibria on a congested transportation network. *SIAM Journal on Algebraic and Discrete Methods*, 2:213–226, 1981.

[3] J. Abara. Applying integer linear programming to the fleet assignment problem. *Interfaces*, 19:20–28, 1989.

[4] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[5] A. Akelik. Travel time functions for transport planning purposes: Davidson's function, its time-dependent form and an alternative travel time function. *Australian Road Research*, 21(3):49–59, 1991.

[6] I. Alder, M.G.C. Resende, and G. Veiga. An implementation of karmarkar's algorithm for linear programming. *Mathematical Programming*, 44:297–335, 1989.

[7] A. Ali and J.L. Kennington. Mnetgen program documentation. Technical report, Dept. of Ind. Eng. and Operations Research, Southern Methodist University, Dallas, 1977.

[8] R. Anbil, C. Barnhart, L. Hatay, and E.L. Johnson. *A column generation technique for the long-haul crew-assignment problem*, volume 2, chapter Optimization in Industry: Mathematical Programming and Optimization Techniques, pages 7–24. T.A. Ciriani and R.C. Leachman (eds.), Willey, 1994.

[9] R. Anbil, E. Gelman, B. Patty, and R. Tanga. Recent advances in crew-pairaing optimization at american airlines. *Interfaces*, 21:62–64, 1991.

[10] A.A. Assad. Solving linear multicommodity flow problems. In *Proceedings of IEEE International Conference on Circuits and Computers, N.G. Rabbat (Ed.)*, volume 1, pages 157–161, 1980.

[11] F. Babonneau, C. Beltran, O. du Merle, C. Tadonki, and J.P. Vial. Generalized analytic center cutting plane method. Technical report, Fond National Suisse and HEC, Univerité de Genève, Genève, 2004.

[12] F. Babonneau, C. Beltran, A. Haurie, C. Tadonki, and J.P. Vial. Proximal-ACCPM : a versatile oracle based optimization method. *Computational Management Science*, To appear, 2006.

[13] F. Babonneau, O. du Merle, and J.P. Vial. Solving large scale linear multicommodity flow problems with an active set strategy and Proximal-ACCPM. *Operations Research*, 54(1):184–197, 2006.

[14] F. Babonneau and J.-P. Vial. Accpm with a nonlinear constraint and an active set strategy to solve nonlinear multicommodity flow problems. *Mathematical Programming*, to appear in 2006.

[15] H. Bar-Gera. Origin-based algorithm for traffic assignment problem. *Transportation Science*, 36(4):398–417, 2002.

[16] C. Barnhart, C.A. Hane, E.L. Johnson, R.E. Marsten, G.L. Nemhauser, and G. Sigismondi. The fleet assignment problem: solving a large scale integer program. *Mathematical Programming*, 70:211–232, 1995.

[17] M. Beckmann, C.B. McGuire, and C.B. Winsten. *Studies in the economics of transpartation*. Yale University Press, 1956.

[18] M. Bellmore, G. Bennington, and S. Lubore. A multivehicle tanker scheduling problem. *Transportation Science*, 5:36–47, 1971.

[19] A. Benchakroun, F. Boyer, and P. Mahey. Capacity and flow assignment of data networks by generalized benders decomposition. *Journal of Global Optimization*, 20:173–193, 2001.

[20] D.P. Bertsekas. Projected newton methods for optimization problems with simple consrtaints. *SIAM Journal Control on Optimization*, 20:221–246, 1982.

[21] D.P. Bertsekas and E.M. Gafni. Projected newton methods and optimization of multicommodity flows. *IEEE Transaction on Automatic and Control*, 28:1090–1096, 1983.

[22] D.P. Bertsekas and R.G. Gallager. *Data Networks*. Prentice-Hall, 1987.

[23] D. Braess. Uber ein paradoxom aus der verkehrsplanung. *Unternehmensfroschung*, 12:258–268, 1968.

[24] D. Branston. Link capacity functions: a rewiew. *Transportation Research*, 10:223–236, 1976.

[25] J. Castro. A specialized interior-point algorithm for multicommodity network flows. *SIAM journal on Optimization*, 10(3):852–877, 2000.

[26] J. Castro. Solving difficult multicommodity flow problems with a specialized interior-point algorithm. *Annals of Operations Research*, 124:35–48, 2003.

[27] J. Castro. Solving quadratic nulticommodity problems through an interior point algorithm. In *System modelling and optimization*, pages 199–212. eds. E.W. Sachs and R. Tichatschke, Kluwer, Boston, 2003.

[28] J. Castro, L. Montero, and D. Rosas. Using ACCPM in a simplicial decomposition algorithm for the traffic assignment problem. Technical report, Statistics and Operations Research Department, Universitat Politecnica de Catalunya, 2002.

[29] J. Castro and N. Nabona. An implementation of linear and nonlinear multicommodity network flows. *European Journal of Operations Research*, 92:37–53, 1996.

[30] P. Chardaire and A. Lisser. Simplex and interior point specialized algorithms for solving nonoriented multicommodity flow problems. *Operations Research*, 50(2):260–276, 2002.

[31] J.R. Correa, A.S. Schulz, and N.E. Stier-Moses. Selfish routing in capacitated networks. *Mathematics of Operations Research*, 29(4):961–976, 2004.

[32] S.C. Dafermos. The traffic assignment problem for multiclass-user transportation networks. *Transportation Science*, 6:73–87, 1972.

[33] S.C. Dafermos. Traffic equilibrium and variational inequalities. *Transportation Science*, 14:42–54, 1980.

[34] S.C. Dafermos. The general multimodal traffic equilibrium problem with elastic demand. *Network*, 12:57–82, 1982.

[35] S.C. Dafermos and F.T. Sparrow. The traffic assignment problem for a general network. *J. Res. National Bureau of standards*, 73B:91–118, 1969.

[36] C.F. Daganzo. On the traffic assignment problem with flow dependent costs I. *Transportation Research*, 11:433–437, 1977.

[37] C.F. Daganzo. On the traffic assignment problem with flow dependent costs II. *Transportation Research*, 11:439–441, 1977.

[38] M. Daneva and P.O. Lindberg. The stiff is moving - conjugate direction Franck-Wolfe methods with applications to traffic assignment. Technical report, Linkoping University, Department of Mathematics, 2004.

[39] G.B. Dantzig and P. Wolfe. The decomposition algorithm for linear programming. *Econometrica*, 29(4):767–778, 1961.

[40] K.B. Davidson. A flow travel time relationship for use in transportation planning. *Proceeding of the Australian Road Research Board*, Part I(3):183–194, 1966.

[41] K.B. Davidson. The theoretical basis of a flow travel time relationship for use in transportation planning. *Australian Road Research*, 8(1):32–35, 1978.

[42] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1:269–271, 1959.

[43] R. Dowling and A. Skabardonis. Improving average travel speeds estimated by planning models. *Transportation Research Record 1366,TRB, National Research Council, Washington D.C.*, pages 68–74, 1992.

[44] J.M. Farvolden, K.L. Jones, I.J. Lustig, and W.B Powell. Multicommodity network flow: The impact of formulation on decomposition. *Mathematical Programming*, 62:95–117, 1993.

[45] J.M. Farvolden, W.B. Powell, and I.J. Lustig. A primal partitioning solution for the arc-chain formulation of a multicommodity network flow problem. *Operations Research*, 41(4):669–693, 1993.

[46] L.K. Fleischer. Approximation fractional multicommodity flow independent of the number of commodities. *SIAM Journal of Discrete Mathematics*, 3:505–520, 2000.

[47] M. Florian, J. Guelat, and H.Spiess. An efficient implementation of the partan variant of the linear approximation method for network equilibrium problem. *Networks*, 17:319–339, 1987.

[48] A. Frangioni. Dual-ascent methods and multicommodity flows. *Ph.D. Thesis TD 5/97, Dip. di Informatica, Univ. di Pisa*, 1997.

[49] A. Frangioni and G. Gallo. A bundle type dual-ascent approach to linear multicommodity min-cost flow problems. *Informs Journal on Computing*, 11:370–393, 1999.

[50] H. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[51] L. Fratta, M. Gerla, and L. Kleinrock. The flow deviation method: an approach to store-and-forward communication network design. *Networks*, 3:97–133, 1973.

[52] E.M. Gafni and D.P. Bertsekas. Two-metric projection methods for constrained optimization. *SIAM Journal on Control and Optimization*, 22(6):936–964, 1984.

[53] B. Gavish and I. Neuman. A system for routing and capacity assignment in computer communication networks. *IEEE Journal on Selected Areas in Communications*, 37:360–366, 1989.

[54] A.M. Geoffrion. Primal resource-directive approaches for optimizing non-linear decomposable systems. *Operations Research*, 18:375–403, 1970.

[55] A.M. Geoffrion and G.W. Graves. Multicommodity distribution systems design by bender's decomposition. *Management Science*, 20:822–844, 1974.

[56] M. Gerla, J.A.S Monteiro, and R. Pazos. Topology design and bandwith allocation in atm nets. *IEEE Journal on Selected Areas in Communications*, 7:1253–1261, 1989.

[57] F. Glineur. Improving complexity of structured convex optimization problems using self-concordant barriers. *European Journal of Operational Research*, 143:291–310, 2002.

[58] J.-L. Goffin, A. Haurie, and J.-P. Vial. Decomposition and nondifferentiable optimization with the projective algorithm. *Management Science*, 38(2):284–302, 1992.

[59] J.-L. Goffin and J.-P. Vial. Convex nondifferentiable optimization: A survey focussed on the analytic center cutting plane method. *Optimization Methods and Software*, 174:805–867, 2002.

[60] J.L. Goffin, J. Gondzio, R. Sarkissian, and J.P. Vial. Solving nonlinear multicommodity flow problems by the analytic center cutting planssae method. *Mathematical Programming*, 76:131–154, 1996.

[61] A.V. Goldberg, J.D. Oldhan, S. Plotkin, and C. Stein. An implementation of a combinatorial approximation algorithm for minimum-cost multicommodity flow. In *Integer Programming and Combinatorial Optimization, Proceedings of the 6th International IPCO Conference, Houston, 1998, R.E. Bixby, E.A. Boyd, R.Z. Rios-Mercado (Eds.)*, pages 338–352, 1998.

[62] B.L. Golden. A minimum cost multicommodity network flow problem concerning imports and exports. *Networks*, 5:331–356, 1975.

[63] J.K. Hartman and L.S. Lasdon. A generalized upper bounding algorithm for multicommodity network flow problems. *Networks*, 1:333–354, 1972.

[64] D. Hearn, S. Lawphongpanich, and S. Nguyen. Convex programming formulations of the asymmetric traffic assignment problem. *Transportation Research*, 18B:357–365, 1984.

[65] D. Hearn, S. Lawphongpanich, and J. Ventura. Finiteness in restricted simplicial decomposition. *Operations Research Letters*, 4:125–130, 1985.

[66] D. Hearn, S. Lawphongpanich, and J. Ventura. Restricted simplicial decomposition: Computation and extenxions. *Mathematical Programming Study*, 31:99–118, 1987.

[67] M. Held, P. Wolfe, and H.P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6:62–88, 1974.

[68] Federal Highway. Urban transportation planning system (utps). *U.S. Dept. of Transportation, Washington, D.C.*

[69] C.A. Holloway. An extension of the frank and wolfe method of feasible directions. *Mathematical Programming*, 6:14–27, 1974.

[70] O. Jahn, R.H Moehring, A.S Schulz, and N.E. Stier-Moses. System-optimal routing flows with user constraints in networks with congestion. *Operations Research*, 53(4):600–616, 2005.

[71] M. Josefsson and M. Patriksson. Sensitivity analysis of separable traffic equilibrium equilibria, with application to bilevel optimization in network design. *report, Department of Mathematics, Chalmers University of technology, Gotheburg*, 2003.

[72] A. Kamath, O. Palmon, and S. Plotkin. Fast approximation algorithms for minimum cost multicommodity flow. In *Proceedings of the Sixth Annual ACN-SIAM Symposium on Discrete Algorithms (SODA), San Fransisco*, pages 493–501, 1995.

[73] J.E. Kelley. The cutting plane method for solving convex programs. *Journal of the SIAM*, 8:703–712, 1960.

[74] J.L. Kennington. Solving multicommodity transportation problems using a primal partitioning simplex technique. *Naval Res. Logist. Quart.*, 24:309–325, 1977.

[75] J.L. Kennington. A survey of linear cost network flows. *Operations Research*, 26:209–236, 1978.

[76] J.L. Kennington and R.V. Helgason. *Algorithm for network programming*. Wiley, New York, 1980.

[77] J.L. Kennington and M. Shalaby. An effective subgradient procedure for minimal cost multicommodity flow problems. *Management Science*, 23:994–1004, 1977.

[78] K.C. Kiwiel. Proximity control in bundle methods for convex nondifferenciable optimization. *Mathematical Programming*, 46:105–122, 1990.

[79] L. Kleinrock. *Communications.* Nets, Stochastic Message Flow and Delay, Dover, 1972.

[80] F.H. Knight. Some fallacies in the interpretation of social cost. *Quart. J. Econom.*, 38:582–606, 1924.

[81] B.W. Kort and D.P. Bertsekas. Combined primal-dual and penalty methods for convex programming. *SIAM Journal on Control and Optimization*, 14:268–294, 1976.

[82] T. Larsson and M. Patriksson. Simplicial decomposition with disaggregated representation for the traffic assignment problem. *Transportation Research*, 26:4–17, 1992.

[83] T. Larsson and M. Patriksson. An augmented lagrangean dual algorithm for link capacity side constrainted traffic assignment problems. *Transportation Research*, 29B:433–455, 1995.

[84] T. Larsson and Di Yuan. An augmented lagrangian algorithm for large scale multicommodity routing. *Computational Optimization and Applications*, 27(2):187–215, 2004.

[85] L. Leblanc. *Mathematical programming algorithms for large scale network equilibrium and network design problems.* PhD thesis, IE/MD Dept, Northwestern University, Evanston IL, 1973.

[86] L. Leblanc, R.V. Helgason, and D.E. Boyce. Improved efficiency of the Franck-Wolfe algorithm for convexs netwok programs. *Transportation Science*, 19:445–462, 1985.

[87] T. Leighton, F. Makedon, S. Plotkon, C. Stein, E. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. *SIAM Journal of Computer and System Sciences*, 50:228–243, 2095.

[88] C. Lemaréchal. An extension of davidon methods to nondifferentiable problems. *Mathematical Programming Study*, 3:95–109, 1975.

[89] C. Lemaréchal. *Nondifferenciable optimization*, volume 1, chapter Optimization, pages 529–572. Handbooks in Operations Research and Management Science, North-Holland, Amsterdam, 1989.

[90] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Mathematical Programming*, 1995.

[91] H.P.L. Luna, P. Mahey, and A. Ouorou. Multicommodity network expansion under elastic demands. *Optimization and Engineering*, 2:277–292, 2001.

[92] P. Mahey, A. Ouorou, L. Leblanc, and J. Chiffet. A new proximal decomposition algorithm for routing in telecommunication networks. *Networks*, 31(4):227–238, 1998.

[93] J.W. Mamer and R.D. McBride. A decomposition-based pricing procedure for large-scale linear program : An application to the linear multicommodity flow problem. *Management Science*, 46:693–709, 2000.

[94] R.D. McBride. Progress made in solving the multicommodity flow problem. *SIAM journal on optimization*, 8(4):947–955, 1998.

[95] A. Nagurney. *Networks Economics: A variational inequality approach*. Advances ib Computational Economics, 1993.

[96] A. Nemirovskii. Lecture notes. *Technion, The Israel Institute of Technology, Faculty of Industrial Engineering and Management*, 1994.

[97] Y. Nesterov. Stable traffic equilibria: properties and applications. *Optimization and Engineering*, 3:29–50, 2000.

[98] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*. Kluwer Academic Publishers, 2004.

[99] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[100] Y. Nesterov and J.-P. Vial. Homogeneous analytic center cutting plane methods for convex problems and variational inequalities. *SIAM Journal on Optimization*, 9(3):707–728, 1999.

[101] Y. Nesterov and J.-P. Vial. Homogeneous analytic center cutting plane methods with approximate centers. *Optimization Methods and Software*, 11-12:243:73, 1999.

[102] A. Ouorou, P. Mahey, and J.P. Vial. A survey of algorithms for convex multicommodity flow problems. *Management Science*, 46:126–147, 2000.

[103] M. Patriksson. *The traffic assignment problem: Models and Methods*. VSP, Utrecht, The Netherlands, 1994.

[104] M. Patriksson. Sensitivity analysis of traffic equilibria. *Transportation Science (to appear)*, 37, 2003.

[105] M. Patriksson and R.T. Rockafellar. Sensitivity analysis of variational inequalities over aggregated polyhedra, with application to traffic equilibria. *Transportation Science*, 37:56–68, 2003.

[106] A.C. Pigou. *The Economics of Welfare*. MacMillan and Co, London, 1920.

[107] L. Portugal, M.G.C Resende, G.Veiga, and J. Judice. A truncated interior-point methodfor the solution of minimum cost flow problems on an undirected multicommodity flow network. In *Proceedings of the First Portuguese National Telecommunications Conference*, pages 381–384, Aveiro, Portugal, 1997.

[108] T. Roughgarden. The price of anarchy is independent of the network topology. *Journal Comput. System Sci.*, 67:341–364, 2003.

[109] T. Roughgarden and E. Tardos. How bad is selfish routing ? *Journal of the ACM*, 49:236–259, 2002.

[110] R.R. Schneur and J.B. Orlin. A scaling algorithm for multicommodity flow problems. *Operations Research*, 46(231-246), 1998.

[111] G.L. Schultz and R.R. Meyer. An interior point method for block angular optimization. *Siam Journal on Optimization*, 1:583–602, 1991.

[112] M. Schwartz and C. Cheung. The gradient projection algorithm for multiple routing in message-switched networks. *IEEE Transaction on Communications*, COM-24:449–456, 1976.

[113] B. Shah, R. Buelher, and O. Kempthorne. Some algorithms for minimizing a function of several variables. *SIAM J. Appl. Math.*, 12:74–92, 1964.

[114] Y. Sheffi. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Models*. Prentice-Hall, New Jersey, 1985.

[115] B. Shetty and R. Muthukrishnan. A parallel projection for the multicommodity flow network. *Journal of the Operational Research Society*, 41:837–842, 1990.

[116] M.J. Smith. The existence, uniqueness and stability of traffic equilibria. *Transportation Research*, 13B:295–304, 1979.

[117] H. Spiess. Conical volume delay functions. *Transportation Science*, 24(2):153–158, 1990.

[118] J.E. Spingarn. Partial inverse of a monotone operator. *Appl. Math. Optim.*, 10:247–265, 1983.

[119] M.A.P. Taylor. Parameter estimation and sensitivity of parameter values in a flow-rate/travel-time relation. *Transportation Science*, 11(3):275–292, 1977.

[120] B. von Holenbalken. Simplicial decomposition in nonlinear programming algorithms. *Mathematical Programming*, 13:49–68, 1977.

[121] J.G. Wardrop. Some theorical aspects of road traffic research. *Proceeding of the institute of civil engineers*, Part II:325–378, 1952.

[122] A. Weintraub, C. Ortiz, and J. Gonzales. Accelerating convergence of the Franck-Wolfe algorithm. *Transportation Research*, 19B:113–122, 1985.

[123] P. Wolfe. Convergence theory in nonlinear programming. In *Integer and Nonlinear Programming*, pages 1–36. J. Abadie (ed.), 1970.