DC programming and DCA for solving some classes of problems in transportation and communication systemes

Anh Son Ta

THÈSE

pour l'obtention du titre de

DOCTEUR DE L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE ROUEN

(arrêté ministériel du 7 août 2006)

Spécialité : MATHÉMATIQUES APPLIQUÉES

Présentée et soutenue par

Anh Son TA

-Titre de la thèse -

Programmation DC et DCA pour la résolution de certaines classes des problèmes dans les systèmes de transport et de communication

Soutenue le 22 Juin 2012

Membres du Jury :

Patrick SIARRY	Rapporteur, Professeur, Université Paris-Est Créteil Val-de-Marne
Nazim AGOULMINE	Rapporteur, Professeur, Université d'Evry Val d'Essonne
Tao PHAM DINH	Directeur de thèse, Professeur, INSA de Rouen
Hoai An LE THI	Directrice de thèse, Professeur, Université de Lorraine
Djamel KHADRAOUI	Responsable scientifique au Public Research Centre Henri Tudor, Luxembourg
Viet Hung NGUYEN	Examinateur, MCF, Université P. & M. Curie
Alain Bui	Examinateur, Professeur, Université de Versailles-St-Quentin-en-Yvelines

THÈSE PRÉPARÉE AU LABORATOIRE DE MATHÉMATIQUES DE L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE ROUEN, FRANCE

REMERCIEMENTS

Dès les premières lignes de ce mémoire, je tiens à remercier tous ceux qui dès le début ont cru en moi et m'ont permis de démarrer et de mener à bien ce travail.

Je souhaiterais exprimer mes remerciements sincères envers le Ministère de l'Éducation Vietnamienne et le Fonds National de la Recherche Luxembourg qui m'ont donné l'opportunité de réaliser cette thèse. Ce travail a été réalisé en collaboration étroit entre le LMI (INSA de Rouen), le LITA (Université de Lorrain) et le CRP Henri Tudor, Luxembourg.

J'exprime ma profonde et respectueuse gratitude à mes directeurs de thèse **Prof. PHAM DINH Tao** et **Prof. LE THI Hoai An** pour leur confiance, leur patience, leur soutien, leur disponibilité et leurs encouragements durant ces années. Merci aussi pour les nombreuses discussions de tout et de rien.

Je tiens à remercier tout particulièrement mon responsable scientifique au CRP Henri Tudor **Dr.Djamel KHADRAOUI**, qui m'a accueilli et m'a encadré dans votre groupe de recherche au CRP Henri Tudor. La confiance que vous m'avez accordée pendant ces quatre années ainsi que votre soutien sans faille m'a permis de progresser rapidement. Soyez assuré de toute mon estime et de mon profond respect.

Je souhaiterais remercier aux rapporteurs : **Prof. Patrick SIARRY** et **Prof. Nazim AGOULMINE** pour leur commentaires et leur questions, tant sur la forme du mémoire que sur son fond, ont contribué à améliorer de manière significative le document.

Mes remerciements s'adressent également à mes collègues qui sont intervenus dans mon travail : *Dr. Phuc, Dr. Thuan, Dr. Quynh, Dr. Minh, M.Sc Cuong* pour les nombreuses discussions que nous avons eu dans les domaines de l'optimisation, programmation et développement.

Je voudrais aussi avoir une pensée pour mes camarades de bureau, par leur présence et leur amitiés, ont partagé leur joie et leur bonne humeur avec moi au sein du laboratoire LMI à Rouen : *Duc Manh, Viet Nga* ; le LITA à Metz : *Bich Thuy, Minh Thuy, Anh Vu* et l'équipe SSI à CRP Henri Tudor : *Moussa, Gérard, Hedi.*

Mes remerciements s'adressent à mes amis Vietnamiens pour tous les bons moments passés ensemble : mes petits frères *Huy Linh, Viet Dung* et mes amis rencontrés à Rouen et à Metz.

Enfin, une énorme pensée à **mes parents, ma famille** et à **Mme. Bach Kim** qui m'ont remonté le moral dans les moments difficiles et les moments d'incertitude. Merci de m'avoir soutenue et de me soutenir encore et toujours.

Dedicated to my little princess Nam!

Résumé

La présente thèse a pour objectif principal de développer des approches déterministes et heuristiques pour résoudre certaines classes des problèmes d'optimisation dans les systèmes de transport et de communication: problèmes de routage, problèmes de covoiturage, problème de contrôle de l'alimentation dans un réseau sans fil, problème d'équilibrage du spectre dans les réseaux DSL. Il s'agit des problèmes d'optimisation non convexe de très grande taille. Nos approches sont basées sur la programmation DC et DCA, méthodes de décomposition proximales et méthodes d'étiquetage des graphes. Grâce aux techniques de formulation/reformulation et de pénalité exacte nous avons établi, de manière simple, des programmes DC équivalents en vue de leur résolution par DCA. En outre, tenant compte de la structure spécifique de ces problèmes, des décompositions DC appropriées et des choix judicieux de points initiaux pour DCA sont proposés afin d'améliorer sa performance. Toutes ces méthodes ont été mises en œuvre avec MATLAB, C/C++. Les simulations numériques montrent la performance de nos algorithmes par rapport à des méthodes existantes, surtout en grande dimension.

Mots-clé : Programmation DC et DCA, Méthode de décomposition proximale, Méthode d'étiquetage des graphes, Problèmes de routage, Problèmes de covoiturage, Problème de contrôle de l'alimentation, Problème d'équilibrage du spectre.

Abstract

In this thesis, we focus on developing deterministic and heuristic approaches for solving some classes of optimization problems in Telecommunication and Mobility & Transport domain: Routing problems, Car pooling problems, Power control problems in wireless network, Optimal spectrum balancing problems in DSL networks. They are large-scale nonconvex optimization problems. Our methodologies focus on DC programming and DCA, Proximal decomposition method and Labeling method in graph theory. They are well-known as powerful tools in optimization. The considered problems were reformulated using the DC formulation/reformulation and exact penalty techniques then the DCA was used to obtain the solution. Also, taking into account the structure of considered problems, we can provide appropriate DC decompositions and the relevant choice strategy of initial points for DCA in order to improve its efficiency. All these proposed methods have been implemented with MATLAB, C/C++ to confirm the practical aspects and enhance our research works.

Keywords: DC programming and DCA, Proximal decomposition method, Labeling algorithms in graph theory, Routing problems, Car pooling problems, Power control problems, Optimal spectrum balancing problems.

Contents

Introduction

"Everything should be made as simple as possible, but not simpler."

Albert Einstein

Over the two last decades, the nonlinear optimization framework has provided both an insightful modeling language and a powerful solution tool to tackle a much wider scope of work in the analysis and design of communication systems. Recently, new demands in the areas of communications and networking, in particular those in the context of mobile network, have derived new challenges for researchers in these domains. So far, a variety of models and methods have been developed, most of them deal with nonconvex optimization framework. A main challenge today is on nonconvex problems in these applications. Several approaches have been proposed in the literature, from nonlinear transformation to turn an apparently nonconvex problem into a convex problem, to characterization of attraction regions and systematically jumping out of a local optimum, from successive convex approximation to dualization, from leveraging the specific structures of the problems (e.g., Difference of Convex functions, concave minimization, low rank nonconvexity) to developing more efficient branch-and-bound procedures.

The works in this thesis are motivated by, on one hand, new challenging problems in the areas of communications and networking, and on another hand, new tools emerging from optimization theory. Such tools include the powerful theories and highly efficient computational algorithms for nonconvex optimization, together with global solution methods and decomposition techniques for solving large scale problems.

Generally speaking, there are two different but complementary approaches for nonconvex programming:

i) Global approaches such as Cutting Plane (CP), Branch and Bound (B&B), Branch and Cut (B&C) can guarantee the globality of the solutions but they are very expensive, in particular for large scale setting;

ii) Local approaches, on the contrary, are much faster while only local minima are available.

Current approaches in communication systems are not generally effective for large scale networks in real world applications. Finding efficient algorithms that realize a compromise between the quality (of solutions), the scalability and the effectively is a challenge of nonconvex programming. Such algorithms must exploit specific structures of the problems being considered. This thesis will address this question while taking into account the following issues in the context of networking and mobile service:

- The topology of networks is dynamic and real-time data transmissions are needed. Hence real-time algorithms are expected.
- Self-organization and self-configuration require all protocols in mobile networks to be distributive and collaborative. By the way, distributed algorithms are necessary.

- Location/Tracking management, in addition to the handover management and routing. In the case of hybrid communication networks, the choice of the best access gateway among a number of available access technologies becomes one of the important considerations. Routing in hybrid networks should be handled by finding a suitable mathematical model and efficient algorithms.

- Multi-user communications service involves large scale setting optimization problems. Therefore the algorithms should be able to solve large size problems.

Our aim is to investigate efficient optimization algorithms meeting these requires. We focus on DC (Difference of Convex functions) Programming and DCA (DC Algorithm), powerful tools and innovative approaches in nonconvex optimization framework. DC programming is an extension of convex programming to cover almost all real world nonconvex optimization problems. As a part of local approaches, DCA has been introduced by Pham Dinh Tao in 1985 and extensively developed by Le Thi Hoai An and Pham Dinh Tao since 1994 to become now classic and increasingly popular (see e.g. [Pham Dinh, 1988]-[Pham Dinh and Le Thi, 1998] and [Le Thi, 1994]-[Le Thi et al., 2002]). DCA aims to solve a general DC program that takes the form

$$\alpha = \inf\{f(x) := g(x) - h(x) : x \in \mathbb{R}^p\} \quad (P_{dc})$$

where g, h are lower semicontinuous proper convex functions on $\mathbb{R}^p$. Such a function f is called DC function, and $g - h$, DC decomposition of f while g and h are DC components of f. DCA is a descent method without line search based on local optimality conditions and DC duality. The construction of DCA involves DC components g and h but not the function f itself: each iteration k of DCA consists of computing

$$y^k \in \partial h(x^k), x^{k+1} \in \arg\min\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle : x \in \mathbb{R}^p\} \quad (P_k).$$

Hence, for a DC program, each DC decomposition corresponds to a different version of DCA. Since a DC function f has an infinite number of DC decompositions which have crucial impacts on the qualities (speed of convergence, robustness, efficiency, globality of computed solutions,...) of DCA, the search for a "good" DC decomposition is important from algorithmic point of views. Moreover, despite its local character, DCA with a good initial point can converge to global solutions. Finding a "good" initial point is then also an important stage of DCA. How to develop an efficient algorithm based on the generic DCA scheme for a practical problem is thus a judicious question to be studied, and the answer depends on the specific structure of the problem being considered.

The use of DCA in this thesis is justified by various reasons:

- DCA is a robust and efficient method for smooth/nonsmooth nonconvex programming which allows to solve large-scale DC programs.

- DCA is simple to use and easy to implement.

- DCA was successfully applied to a lot of different and various nonconvex optimization problems to which it quite often gave global solutions and proved to be more robust and more efficient than related standard methods. In particular, DCA has already efficiently solved some DC programs in network optimization (see [Le Thi and Pham Dinh, 2002], [Le Thi et al., 2008a], [Le Thi et al., 2008b]) and transportation systems ([Le Thi et al., 2008]...).

- By exploiting the nice effect of DC decomposition of the objective function we can design distributed algorithms. This issue is very important in communication networks that involve multi-users, in particular in the purpose of personalized mobile services.

The objective of this thesis is to investigate DC programming and DCA to some important classes of problems in communication systems and transportation systems, in the context of mobility networks. The following areas in communications systems are studied:

- Routing: several models, from Unicast QoS routing to the complex Multicast and Many to Many Multicast QoS routing, are considered. The traditional models of routing in networks are those of Multicommodity Network Optimization Problems which are often in the form of mixed integer programming (continuous/binary and/or integer variables). In this thesis, we tackle the routing problems by developing the approaches adapted to dynamic networks and based on DCA for Mixed Linear Integer Programming (MLIP) models and proximal decomposition techniques (for large scale problems). We also consider a so called Partitioning-Hub Location-Routing Problem (PHLRP) that consists of partitioning a given network into sub-networks, locating at least one hub in each sub-network, and routing the traffic within the network while minimizing the cost.

- Power control: power control and resource allocation techniques for cellular communication systems have been a recent focus of intensive studies. It has been proposed to use the user signal to interference plus noise ratio (SINR) to adjust the transmitted power. Several schemes for power control, centralized or distributed, have been extensively studied since 1990s based on different transmission models and application needs. Various objectives have been considered for developing power control algorithms. Particularly, one can maximize the minimum SINR, minimize total transmitted power, or minimize outage probability in a cellular network [Chiang, 2006a]. In this thesis, we consider new approaches of formulating power control problems through DC programming framework, and develop efficient DCA schemes for the new DC formulation.

- Dynamic Spectrum Management (DSM): the DSM is an effective technique for mitigating detrimental effect of crosstalk in Digital Subscriber Lines (DSL). Among various DSM techniques, centralized Optimal Spectrum Balancing (OSB) achieves the maximum possible data rates by computing the optimal PSDs for all modems in DSL systems. Unfortunately, its computational complexity grows exponentially in the number of users N and becomes intractable for large N. To reduce the complexity of OSB, we will exploit the fact that the non-convex optimization problem in OSB can be reformulated as an equivalent global concave minimization problem by representing its objective function explicitly as a DC function. Hence we can investigate new and efficient algorithms based on DC programming and DCA for solving nonconvex optimization problems in OSB.

Also in the context of mobility network, we study one interesting application in transportation systems, the Car Pooling problem. Car pooling consists of managing the sharing of a pool of cars between several users that have whole or part of their route in common. To solve the problem, several tasks should be performed: choosing drivers and passengers, allocating passengers to cars, computing an optimized route for the cars. In this thesis, we tackle the Car pooling problem by two approaches: in the first approach the Car pooling problem is formulated as a Mix Integer Linear Program (MILP) for which DCA and the combined DCA- Branch and Bound are investigated. In the second approach we consider the Car pooling as a multiobjective programming problem and develop a heuristic algorithm for solving it.

From a mathematical point of view we can classify the problems studied in our work in three categories:

i) Binary Integer Linear Program (BILP) or Mixte Integer Linear Program (MILP): it concerns with Unicast QoS routing, complex Multicast and Many to Many Multicast

QoS routing (BILP), Partitioning-Hub Location-Routing, and the Car Pooling problem (MLIP).

ii) Continuous nonconvex programming: this category contains the Power control problem and the DSM problem.

iii) Multiobjective programming problem: it deals with the Car pooling problem.

On the methodology, the use of DCA is universal in our work: we investigate DCA for solving all considered problem. Indeed, although DCA is a continuous approach, its can be used for solving the MILP via an exact penalty technique. DCA has been already used for solving MILP in several applications (see e.g. [Pham Dinh et al., 2010], [Le Thi and Pham Dinh, 2001], [Le Thi et al., 2007],...) In this work, exploiting the special structure of problems being considered, we adapt DCA for the resolution of our problems by developing several techniques: decomposition proximal techniques for QoS routing to get the scalability of DCA, combination between DCA and Branch and Bound algorithms to ensure the globality of solutions, dynamic update penalty parameter for obtaining good initial points of DCA and the efficiency of DCA, ... We compare DCA with CPLEX, the best solver for MILP. The comparative numerical results prove the efficiency, the scalability and the superiority of DCA with respect to CPLEX, especially in large scale setting problems.

It is interesting to note that although DCA works in a continuous domain, it gives an integer solution with an appropriate value of penalty parameter.

The two (continuous) hard nonconvex programs (the Power control problem and the DSM problem) are DC programs. It is easy to derive natural DC decompositions from the definition of objective functions. But from a computational point of view, we observe that other DC decompositions may be more *interesting*. In this work we propose *elegant* DC decompositions for these problems that give birth very inexpensive DCA schemes. Numerical experiments show that our algorithms are more efficient than the best existing algorithms.

The thesis is divided into four parts:

- The first part deals with the background of DC programming and DCA, and the basis notations/algorithms. After a presentation of the basics in the DC programming and DCA in Section 1.2, we explore the technique of the Branch and Bound and the DCA-Branch and Bound for solving Mixed 0-1 Linear Program in Section 1.3. Section 1.4 dedicates to a short presentation of the Proximal Decomposition Method in convex programming.

- The second part is devoted to solve the Routing Problems:

  - In Chapter 2, we study the QoS routing problems, from classical and simple Unicast QoS routing problems to the complex Multicast and Many to Many Multicast QoS routing problems. The problems are first formulated as Concave Quadratic Program (CQP) or Binary Integer Linear Programs (BILP). Then we investigate DC programming and DCA for solving them. Finally, we tackle large scale setting problems (with large numbers of variables and constraints) by using the Proximal Decomposition Method to solve sub-convex programs at each iteration of DCA.
  - In Chapter 3, a specific case of the hub location problem called Partitioning-Hub Location-Routing Problem (PHLRP) is considered. Basing on the formulation given in [Ozsoy et al., 2008], we improve it and propose a new formulation on which DC programming and DCA are applied.

- The third part is dedicated to the solution of the Power control problems in wireless network (Chapter 4) and the Optimal spectrum balancing problems in DSL network (Chapter 5) by DC programming and DCA.

- The last part addresses the Car Pooling Problems. We first introduce the Car pooling problem and then study two approaches for solving them. The deterministic approach based on DC programming and DCA and Branch and Bound is presented in Chapter 6. The classical car pooling problem is formulated as a MILP for which DCA and the combined DCA- Branch and Boubd are investigated. Chapter 7 introduces the multiobjective car pooling model, which is an extension of the classical problem. Based on labeling algorithms for solving multiobjective shortest path problem, we introduce a new heuristic algorithm for solving the new problem.

Publications

Journal papers

[1]. A. S. Ta, H.A. Le Thi, D. Khadraoui, T. Pham Dinh, *Solving Partitioning-Hub Location-Routing Problem using DCA*, Journal of Industrial and Management Optimization (JIMO), vol.8, Issue.1, 2012, pp.87-102.

Conference with review committee

[2]. A. S. Ta, H.A. Le Thi, D. Khadraoui, T. Pham Dinh, *Solving QoS Routing Problems by DCA*, In Proc. 2nd ACIIDS, Intelligent Information and Database Systems, Lecture Notes in Artificial Intelligence (LNAI), pages 460-470, Hue, Vietnam, 2010. Springer Verlag 5991.

[3]. A. S. Ta, H.A. Le Thi, D. Khadraoui, T. Pham Dinh, *Solving Multicast QoS Routing Problem in the context V2I Communication Services using DCA*, 9th IEEE/ACIS (ICIS 2010), August 18-20, 2010, Yamagata, Japan, pp.471-476 (published by IEEE Xplore).

[4]. A. S. TA, H. A. Le Thi, T. Pham Dinh, *Power control by DC programming and DCA*, Proceedings of International Conference on Industrial Engineering and Systems Management IESM 2011, May 25-27, 8 pages.

[5]. A. S. Ta, H. A. Le Thi, G. Arnould, D. Khadraoui, T. Pham Dinh, *Solving Car Pooling Problem using DCA*, Proceeding of Global Information Infrastructure Symposium (GIIS 2011), Danang 4-6/August/2011, 6 pages (published by IEEE Xplore).

[6]. A.S Ta, H. A Le Thi, T. Pham Dinh, and D. Khadraoui. *Solving many to many multicast qos routing problem using dca and proximal decomposition technique*, In Proc. International Conference on Computing, Networking and Communications, pages 809-814, Hawaii, American, 30/January-2/February, 2012. (published by IEEEXplore.)

[7]. A.S Ta, H. A Le Thi, T. Pham Dinh, and D. Khadraoui. *A distributed algorithm solving multiobjective dynamic car pooling problem.* Proc. International Conference on Computer and Informatic Science, 11-14/June, Kuala Lumpur, Malaysia, 2012. (published by IEEEXplore)

Communication conference

[8]. A. S. Ta, H. A. Le Thi, T. Pham Dinh, T. Le Ngoc, *Optimal spectrum balancing in multi-user DSL network by DC programming and DCA*, invited session on Novel opportunities of DC programming and DCA for Industry and Finance, 23rd European Conference on Operational Research, Bonn, July 5 - 8, 2009. (abstract)

[9]. A. S. Ta, H.A. Le Thi, D. Khadraoui, T. Pham Dinh, *Solving Partitioning-Hub Location-Routing Problem using DCA*, Proceeding of The 8th International Conference on Optimization: Techniques and Applications, Shanghai 12/2010. (abstract).

List of abbreviations

$\chi_C(.)$ the indicator function of C

$\Gamma_0(X)$ the set of all l.s.c proper convex functions on X

$\langle ., . \rangle$ the scalar product

$\text{Proj}_C(.)$ the projection operator on the set C

$\partial f(x)$ the subdifferential of f at point x

$f^\star$ the conjugate function of f

$Gr(T)$ the graph of operator T

$SINR$ Signal to Interference plus Noise Ratio

X the Euclidean space IR^n

Y the dual vector space of X

ADSL Asymmetric digital subscriber line

AWGN Additive white Gaussian noise

B&B Branch and Bound

BBDCA DCA combined with classical Branch and Bound algorithm

BILP Binary Integer Linear Program

$\text{Co}(X)$ the set of convex functions on X

CPLEX an efficient software for solving optimization problems

CPP car pooling problem

CQP Concave Quadratic Program

DC Difference of Convex functions

$\text{DC}(\Omega)$ the set of all DC functions on Ω

DCA DC algorithms

DMT Discrete multitone

$\text{dom}(f)$ the effective domain of a function f

DSL Digital Subscriber Lines

DSM Dynamic spectrum management

$\text{epi}(f)$ the epigraph of a function f

ISB Iterative Spectrum Balancing

ITU International Telecommunication Union

IWF Distributed Iterative Water Filling

l.s.c lower semi-continuous

MCM Multi-Constrained Mutlicast Tree

MCOM Multi-Constrained Optimal Mutlicast Tree

MCOP Multi-Constrained Optimal Path

MCP Multi-Constrained Path

MCPP multiobjcetive car pooling problem

MILP0-1 Mix 0-1 integer linear programming

OSB Optimal Spectrum Balancing

PDM proximal decomposition method

PHLRP Partitioning-Hub Location-Routing Problem

PSD Power spectral density

QoS Quality of Service

SCALE Successive Convex Approximation for Low complExity

SIW Selective Iterative Water Filling

VDSL Very-high-bit-rate digital subscriber line

VRPPD Vehicle Routing Problem with Pickups and Deliveries

Part I

PRELIMINARY STUDIES

Chapter 1

Preliminary Studies

1.1 Introduction

We report in this chapter a brief presentation of preliminary studies about DC programming, DCA, Proximal Decomposition Method and how to use DCA, DCA combined with Branch and Bound technique for solving mixed 0-1 integer linear program, that we will be most useful in the sequel. This chapter is organized as follows. In section 1.2, we present the main results concerning DC programming and DCA . The detailed content was introduced in [Pham Dinh and Le Thi, 1997] and [Le Thi, 1994]. Section 1.3 address methods based on DC programming and DCA for solving mixed 0-1 integer linear programming (MILP0-1). After that, we combine Branch and Bound (B&B) and DCA algorithms to globally solve these problems. Section 1.4 provides a brief description of Proximal Decomposition Method which is applied to solve convex optimization problems, that will be used to combine with DCA to handle large scale problems.

1.2 DC programming and DCA

The framework of *convex programs* is proved too narrow, and the notion of convex function has succeeded with that happiness, more general, DC function (difference of convex functions). DC functions have many important properties that were derived from 1950s by Alexandroff (1949), Landis (1951) and Hartman (1959). One of the main properties is their relative stability in operations frequently used in optimization. However, it was until the mid-80s when the class of DC functions was introduced in optimization, and expanding the class of optimization problems with the appearance of DC programming. There are two different but complementary approaches, we can say two schools, in DC programming:

1. The combinatorial approach (this terminology is due to the fact that new introduced tools were inspired by the concepts of combinatorial optimization) in continuous global optimization,

2. The analysis convex approach in nonconvex optimization.

The first approaches are developed according to the spirit of the combinatorial optimization, but with the difference that one works in the continuous frame work. In these approaches, optimal solutions are located by using the approximation methods, for instance, cutting techniques, decomposition methods, branch and bound, etc. The pioneer of this approach is H. Tuy whose first work was introduced in 1964. His work is abundant, included books by Horst-Tuy ([Tuy, 1993, 1995]) which present the theory, algorithms and applications of global optimization. Among the most important contributions to this approach, it is worth citing the ones by Hoang Tuy, R. Horst, H. Benson, H. Konno, P. Pardalos, Le Dung Muu, Le Thi Hoai An, Nguyen Van Thoai, Phan Thien Thach and Pham Dinh Tao. However, the relatively sophisticated algorithms are very expensive to implement, then they should be reserved for problems of reasonable size with appropriate structures to methods when it is important to locate global optimum. Therefore, they definitively do not achieve the aspiration of solving real life programs in their true dimension.

The second approach relies on the powerful arsenal of analysis and convex optimization. The first work, due to Pham Dinh Tao (1975), concerning the calculation of matrix norms (fundamental problem in numerical analysis) that is a problem of maximizing a convex function over a convex set. The work of Toland (1978) [Toland, 1978] on the duality and optimality local DC optimization generalizes elegantly the results established by Pham Dinh Tao in convex maximization. The DC optimization theory is developed by Pham Dinh Tao, J. B. Hiriart Urruty, Jean - Paul Penot, Phan Thien Thach, Le Thi Hoai An. On the algorithmic part of the second approach, currently available as DCA (DC Algorithms) introduced by Pham Dinh Tao (1986), which are based on optimality conditions and duality in DC optimization. But it took until the joint work of Le Thi Hoai An and Pham Dinh Tao (see [Le Thi, 1994]-[Le Thi et al., 2002] and [Pham Dinh, 1988]-[Pham Dinh and Le Thi, 1998]) to show that it definitely needed in nonconvex optimization as one of the simplest and most performance algorithms, capable of handling large problems.

We report in this section the main results of DC Programming and DCA which will be most useful for our work. These results are extracted from those presented in H. A. Le Thi 1994 ([Le Thi, 1994]), H. A. Le Thi 1997 ([Le Thi, 1997]). For a detailed discussion we refer to these two references (see also [Le Thi, 1994]-[Le Thi et al., 2002] and [Pham Dinh, 1988]-[Pham Dinh and Le Thi, 1998]).

1.2.1 Fundamentals of DC analysis

Notations and properties

This paragraph is devoted to a brief recall of convex analysis for facilitating the reading of certain passages. For more details, we refer to the work of P.J Laurent [Laurent, 1972], of R.T Rockafellar [Rockafellar, 1970] and of A. Auslender [Auslender, 1976]. Let X be the Euclidean space $\mathbb{R}^n$, $\langle .,. \rangle$ be the scalar product, $\|x\| = \langle x, x\rangle^{\frac{1}{2}}$ be the Euclidean norm, and the dual vector space of X is denoted by Y, which can be identified with X itself. We use an usual tool of convex analysis where a function can take the infinite value $\pm\infty$ [Rockafellar, 1970]. We note $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$. A function $f : S \longrightarrow \overline{\mathbb{R}}$ is defined on a convex set S in X, the effective domain of f, denoted by $\mathrm{dom}(f)$, is

$$\mathrm{dom}(f) = \{x \in S : f(x) < +\infty\} \tag{1.1}$$

and the epigraph of f, denoted by $\mathrm{epi}(f)$, is

$$\mathrm{epi}(f) = \{(x, \alpha) \in S \times \mathbb{R} : f(x) < \alpha\}.$$

If $\mathrm{dom}(f) \neq \emptyset$ and $f(x) > -\infty$ for all $x \in S$ then we say that the function $f(x)$ is *proper*.

A function $f : S \longrightarrow \overline{\mathbb{R}}$ is called *convex* if its epigraph is a convex set in $X \times \overline{\mathbb{R}}$. This is equivalent to saying that S is a convex set in X and for all $\lambda \in [0, 1]$ we have

$$f((1-\lambda)x^1 + \lambda x^2) \leq (1-\lambda)f(x^1) + \lambda f(x^2) : \forall x^1, x^2 \in S. \tag{1.2}$$

Let $\mathrm{Co}(X)$ be the set of convex functions on X.

In (1.2), if the strict inequality holds for all $\lambda \in]0, 1[$ and for all $x^1, x^2 \in S$ with $x^1 \neq x^2$ then f is called *strictly convex* function.

A function f is called *strongly convex* on a convex set C if there exists a number $\rho > 0$ such that

$$f((1-\lambda)x^1 + \lambda x^2) \leq (1-\lambda)f(x^1) + \lambda f(x^2) - (1-\lambda)\lambda\frac{\rho}{2}\|x^1 - x^2\|^2, \tag{1.3}$$

for all $x^1, x^2 \in C$, and for all $\lambda \in [0,1]$. It is amount to saying that $f - \frac{\rho}{2}\|.\|^2$ is convex on C. The *modulus of strong convexity* of f on C, denoted by $\rho(f,C)$ or $\rho(f)$ if $C = X$, is given by

$$\rho(f,C) = Sup\{\rho \geq 0 : f - \frac{\rho}{2}\|.\|^2 \text{ is convex on } C\} > 0. \tag{1.4}$$

Clearly, f is convex on C if and only if $\rho(f,C) \geq 0$. One says that f is *strongly convex* on C if $\rho(f,C) > 0$.

Remark 1.1 *f strongly convex $\Longrightarrow$ f strictly convex $\Longrightarrow$ f convex.*

Let f be a proper convex function on X, a vector $y^0 \in Y$ is called a *subgradient* of f at a point $x^0 \in \text{dom}(f)$ if

$$\langle y^0, x - x^0\rangle + f(x^0) \leq f(x) \quad \forall x \in X.$$

The set of all subgradients of f at x^0 is called the *subdifferential* of f at x^0 and is denoted by $\partial f(x^0)$.

Let $\varepsilon > 0$, a vector y^0 is called ε-subgradient of f at point x^0 if

$$\langle y^0, x - x^0\rangle + f(x^0) \leq f(x) + \varepsilon \quad \forall x \in X.$$

Then the set of all $\varepsilon - subgradients$ of f at point x^0 is called the the ε-subdifferential of f at x^0 and is denoted by $\partial_\varepsilon f(x^0)$.

A function $f : S \longrightarrow \mathbb{R}$ is called *lower semi-continuous* (l.s.c) at a point $x \in S$ if

$$\liminf_{y \to x} f(y) \geq f(x).$$

Let $\Gamma_0(X)$ be the set of all l.s.c proper convex functions on X.

Definition 1.1 *Let a function $f : X \longrightarrow \mathbb{R}$, the* conjugate function *$f^\star$ of f, is a function belonging to $\Gamma(Y)$ and defined by*

$$f^\star(y) = \sup\{\langle x, y\rangle - f(x) : x \in X\}. \tag{1.5}$$

$f^\star$ is an upper envelope of continuous affine functions $y \mapsto \langle x, y\rangle - f(x)$ on Y.

The main properties are summarized in the following proposition that will be needed for further:

Proposition 1.1 *If $f \in \Gamma_0(X)$ then:*

- *$f \in \Gamma_0(X) \Longleftrightarrow f^\star \in \Gamma_0(Y)$. In this case, we have $f = f^{\star\star}$,*
- *$y \in \partial f(x) \Longleftrightarrow f(x) + f^\star(y) = \langle x, y\rangle$ and $y \in \partial f(x) \Longleftrightarrow x \in \partial f^\star(y)$,*
- *$\partial f(x)$ is a closed convex set,*
- *If $\partial f(x) = \{y\}$ then f is differentiable at x and $\nabla f(x) = y$,*
- *$f(x^0) = min\{f(x), x \in X\} \Longleftrightarrow 0 \in \partial f(x^0)$.*

Polyhedral convex functions

A convex set C is called a *polyhedral convex* if

$$C = \bigcap_{i=1}^{m} \{x : \langle a_i, x\rangle - \alpha_i \leq 0\} \text{ where } a_i \in Y, \alpha_i \in \mathbb{R}, \quad \forall i = 1, ..., m.$$

A function is called a polyhedral convex if

$$f(x) = \sup\{\langle a_i, x\rangle - \alpha_i : i = 1, ..., k\} + \chi_C(x).$$

where C is a polyhedral convex set and $\chi_C(.)$ stands for the indicator function of C, i.e. $\chi_C(x) = 0$ if $x \in C$ and $+\infty$ otherwise.

Proposition 1.2 *([Rockafellar, 1970])*

- *Let f be a polyhedral convex function. f is everywhere finite if and only if $C = X$,*
- *f is polyhedral convex then $f^\star$ is also polyhedral. Moreover, if f is everywhere finite then*

$$f(x) = \sup\{\langle a_i, x\rangle - a_i : i = 1, ..., k\},$$

$$dom(f^\star) = co\{a_i : i = 1, ..., k\},$$

$$f^\star(y) = \min\{\Sigma_{i=1}^k \lambda_i \alpha_i : y = \Sigma_{i=1}^k \lambda_i a_i, \lambda_i \geq 0, \Sigma_{i=1}^k \lambda_i = 1\},$$

- *If f is polyhedral then $\partial f(x)$ is a nonempty polyhedral convex set at every point $x \in dom(f)$.*

DC functions

A function $f : \Omega \mapsto \overline{\mathbb{R}}$ defined on a convex set convex $\Omega \subset \mathbb{R}^n$ is called DC on Ω if it can be presented in the form of difference of two convex functions on Ω, i.e.

$$f(x) = g(x) - h(x),$$

where g and h are convex functions on Ω, $g - h$ is called a DC decomposition of f. Let $DC(\Omega)$ be the set of all DC functions on Ω, and $DC_f(\Omega)$ in case of g and h are finite convex on Ω.

DC functions have many important properties that were derived from 1950s by Alexandroff (1949), Landis (1951) and Hartman (1959); one of the main properties is their stability with respect to frequently used operations in optimization. Specifically:

Proposition 1.3 *(i) A linear combination of DC functions on Ω is DC on Ω,*

(ii) The upper envelope of a finite set of finite DC functions on Ω is DC on Ω,
The lower envelope of a finite set of finite DC functions on Ω is DC on Ω,

(iii) Let $f \in DC_f(\Omega)$, then $|f(x)|, f^+(x) = max\{0, f(x)\}$ and $f^-(x) = min\{0, f(x)\}$ are DC on Ω.

These results generalize to the case of value in $\mathbb{R} \cup \{+\infty\}$ ([Le Thi, 1997]). It follows that the set of DC functions on Ω is a vector space ($DC(\Omega)$): it is the smallest vector space containing all convex functions on $\Omega(Co(\Omega))$.

Remark 1.2 *Given a DC function f and a DC decomposition $f = g - h$, then for any finite convex function φ, $f = (g+\varphi) - (h+\varphi)$ gives another DC decomposition of f. Thus, a DC function has an infinite number of DC decompositions.*

Denoted by $C^2(\mathbb{R}^n)$, the class of twice continuously differentiable functions on $\mathbb{R}^n$.

Proposition 1.4 *Any function $f \in C^2(\mathbb{R}^n)$ is DC on an arbitrary compact convex set $\Omega \subset \mathbb{R}^n$.*

Since the subspace of polynomials on Ω is dense in the space $C(\Omega)$ of continuous functions on Ω, we have:

Corollary 1.1 *The space of DC functions on a compact convex set $\Omega \subset \mathbb{R}^n$ is dense in $C(\Omega)$, i.e.*

$$\forall e > 0, \exists F \in C(\Omega) : |f(x) - F(x)| \leq \varepsilon \quad \forall x \in \Omega.$$

Note that DC functions occur very frequently in practice, both differentiable and non-differentiable optimization. An important result established by Hartman (1959) permits identified DC functions in many situations, simply by using a local analysis of the convexity (local convex, local concave and local DC).

A function $f : D \mapsto \mathbb{R}$ defined on an open convex set $D \in \mathbb{R}^n$ is called local DC if for all $x \in D$ there is an open convex neighborhood U of x and a pair of convex functions g, h on U such that $f|_U = g|_U - h|_U$.

Proposition 1.5 *A local DC function on a convex set D is DC on D.*

1.2.2 DC optimization

Due to the preponderance and wealthy properties of DC functions, the transition of the subspace $Co(\Omega)$ to the vector space $DC(\Omega)$ permits to expand significantly convex optimization problems in the non-convexity. The field of optimization problems involving DC functions is relative large and open, covering most of the problems encountered in applications.

However, we can not immediately deal with any non-convex and non-differentiable optimization problem. The following classification has now become classic:

$$(1) \quad \sup\{f(x) : x \in C\}, \text{where f and C are convex}$$

$$(2) \quad \inf\{g(x) - h(x) : x \in X\}, \text{ where g and h are convex}$$

$$(3) \quad \inf\{g(x) - h(x) : x \in C, f_1(x) - f_2(x) \leq 0\},$$

where g, h, f_1, f_2 and C are convex, these seem to be large enough to contain substantially all nonconvex problems encountered in everyday life. Problem (1) is a special case of Problem (2) with $g = \chi_C$, the indicator function of C and $h = -f$. Problem (2) can be modified in the form equivalent to (1)

$$\inf\{t - h(x) : g(x) - t \leq 0\}.$$

While Problem (3) can be transformed to the form (2) by using exact penalty related to the DC constraints $f_1(x) - f_2(x) \leq 0$. Its resolution can also be reduced under certain technical conditions, that is a series of Problems (1). Problem (2) is called *a DC program*. It is a major interest both from practical and theoretical point of view. From the

theoretical point of view, we can note that, as we noted above, the class of DC functions is remarkable stable with the operations frequently used in optimization. Moreover, there is an elegant duality theory ([Pham Dinh, 1975, 1976, Toland, 1978, Urruty, 1985, Le Thi, 1994, 1997, Le Thi and Pham Dinh, 1997]) which, as convex optimization, has profound practical implications for numerical methods.

DC algorithms (DCA) is introduced by Pham Dinh Tao ([Pham Dinh, 1986, 1988]) who presented a new approach based on the DC theory. In fact, these algorithms are a generalization of subgradients algorithms which were studied by the same author on the convex maximization ([Pham Dinh, 1975, 1986]). However, it was until the joint work of Le Thi et Pham Dinh during the past decades (see [Le Thi, 1994]-[Le Thi et al., 2002] and [Pham Dinh, 1988]-[Pham Dinh and Le Thi, 1998])) that DCA has now become classical and popular.

DC duality

In convex analysis, the concept of duality (conjugate function, dual problem, etc.) is a very powerful fundamental concept. For convex problems and in particular linear, a duality theory has been developed over several decades [Rockafellar, 1970]. More recently, an important concept of duality in nonconvex analysis has been proposed and developed, first for convex maximization problems, before reaching the DC problems. DC duality introduced by Toland (1978) can be regarded as a generalization of logic work of Pham Dinh Tao (1975) on convex maximization. We will present below the main results (in DC optimization) on optimal conditions (local and global) and the DC duality. For more details, the reader is referred to the document of Le Thi (1997) (see [Le Thi and Pham Dinh, 1997]).

Let space $X = \mathbb{R}^n$, usual inner product $\langle ., . \rangle$ and the Euclidean norm $\|.\|$. Let Y be the dual space of X which can be identified with X itself and $\Gamma_0(X)$ be the set of all proper l.s.c convex functions on X.

Given $g(x)$ and $h(x)$ are two proper convex functions on X $(g, h \in \Gamma_0(X))$, considering the DC problem

$$\inf\{g(x) - h(x) : x \in X\} \quad (P)$$

and the dual problem

$$\inf\{h^\star(y) - g^\star(y) : y \in Y\} \quad (D)$$

where $g^\star(y)$ (resp. $h^\star(y)$) denotes the conjugate function of g (resp. h).

The results of DC duality defined by using the conjugate functions give an important relationship in DC optimization [Toland, 1978].

Theorem 1.1 *Let g and $h \in \Gamma_0(X)$, then*

(i)

$$\inf_{x \in dom(g)} \{g(x) - h(x)\} = \inf_{y \in dom(h^\star)} \{h^\star(y) - g^\star(y)\} \tag{1.6}$$

(ii) If y^0 is a minimizer of $h^\star - g^\star$ on Y then $x^0 \in \partial g^\star(y^0)$ is a minimizer of $g - h$ on X.

Proof 1.1 *(i)*

$$\begin{aligned}\alpha &= \inf\{g(x) - h(x) : x \in X\}\\ &= \inf\{g(x) - \sup\{\langle x, y\rangle - h^\star(y) : y \in Y\} : x \in X\}\\ &= \inf\{g(x) + \inf\{h^\star(y) - \langle x, y\rangle : y \in Y\} : x \in X\}\\ &= \inf_x \inf_y\{h^\star(y) - \langle x, y\rangle - g(x)\}\\ &= \inf\{h^\star(y) - g^\star(y) : y \in Y\}.\end{aligned}$$

(ii) cf. Toland ([Toland, 1978]).

The theorem 1.1 shows that solving the primal problem (P) implies resolution of the dual problem D and vice versa.

Global optimality in DC optimization

In convex optimization, x^0 minimizes a function $f \in \Gamma_0(X)$ if and only if $0 \in \partial f(x^0)$. In DC optimization, the following global optimality conditions [Urruty, 1989] are formulated by using ε-subdifferential of g and h. His demonstration is based on studying the behavior of ε-subdifferential of a convex function depending on the parameter ε. The demonstration in [Le Thi, 1997] is more simple and suitable in case of DC optimization: it simply expresses that global optimality condition is a geometry translation and optimal values of primal and dual DC programs are equal.

Theorem 1.2 ***(Global DC optimization)*** *Let $f = g - h$ where $g, h \in \Gamma_0(X)$ then x^0 is a global minimizer of $g(x) - h(x)$ on X if and only if,*

$$\partial_\varepsilon h(x^0) \subset \partial_\varepsilon g(x^0) \quad \forall \varepsilon > 0. \tag{1.7}$$

Remark 1.3

(i) If $f \in \Gamma_0(X)$, we can write $f = g - h$ with $f = g$ and $h = 0$. In this case, the global optimal in (P) - which is the same as the local optimal in (P) (because (P) is a convex problem) - is characterized by,

$$0 \in \partial f(x^0). \tag{1.8}$$

The fact that $\partial_\varepsilon h(x^0) = \partial h(x^0) = \{0\}, \quad \forall \varepsilon > 0, \forall x \in X$, the relationship (1.8) is equivalent to (1.7).

(ii) More generally, considering DC decompositions of $f \in \Gamma_0(X)$ in the form $f = g - h$ with $g = f + h$ and $h \in \Gamma_0(X)$ finite everywhere on X. The corresponding DC problem is a "false" DC problem because it is a convex optimization problem. In this case, the relationship (1.8) is equivalent to

$$\partial h(x^0) \subset \partial g(x^0).$$

(iii) We can therefore say that (1.7) clearly marks the transition from convex optimization to nonconvex optimization. This feature of the global optimality of (P) indicates the complexity of its practical use because it appeals to all ε-subdifferential at x^0.

Local optimality in DC optimization

We have seen that the relationship $\partial h(x^0) \subset \partial g(x^0)$ (using the subdifferential "exact") is necessary and sufficient condition of global optimization for a "false" DC problem (a convex optimization problem). In a global optimization problem, the minimization function is local convex "around" a local minimum, then it is clear that the relation of subdifferential inclusion will characterize a local minimum of a DC problem.

Definition 1.2 *Let g and $h \in \Gamma_0(X)$. A point $x^\bullet \in dom(g) \cap dom(h)$ is a local minimizer of $g(x) - h(x)$ on X if and only if*

$$g(x) - h(x) \geq g(x^\bullet) - h(x^\bullet), \quad \forall x \in V_{x^\bullet}, \tag{1.9}$$

where V_x denotes a neighborhood of x.

Proposition 1.6 ***(Necessary condition of local optimality)*** *If $x^\bullet$ is a local minimizer of $g - h$ then*

$$\partial h(x^\bullet) \subset \partial g(x^\bullet). \tag{1.10}$$

Proof 1.2 *If $x^\bullet$ is a local minimizer of $g - h$, then there exists a neighborhood V_x of x such that*

$$g(x) - g(x^\bullet) \geq h(x) - h(x^\bullet), \quad \forall x \in V_{x^\bullet}. \tag{1.11}$$

Therefore if $y^\bullet \in \partial h(x^\bullet)$ then

$$g(x) - g(x^\bullet) \geq \langle x - x^\bullet, y^\bullet \rangle, \quad \forall x \in V_{x^\bullet} \tag{1.12}$$

which is equivalent, under the convexity of g, at $y^\bullet \in \partial g(x^\bullet)$.

Note that for a number of DC problems and in particular for h polyhedral ones, the necessary condition (1.10) is also sufficient, as we will see a little further. We say that $x^\bullet$ is a critical point of $g - h$ if $\partial h(x^\bullet) \cap \partial g(x^\bullet)$ is non empty [Toland, 1978]. It is a weakened form of subdifferential inclusion. The search for such a critical point is at the DCA (simple form) which will be studied in the next section. In general, DCA converges to a local solution of a DC optimization problem. However, in theory, it is important to formulate sufficient conditions for local optimality.

Theorem 1.3 ***(Sufficient condition of local optimality ([Le Thi, 1997, Le Thi and Pham Dinh, 1997]))*** *If $x^\star$ admits a neighborhood V such that*

$$\partial h(x) \cap \partial g(x^\star) \neq \emptyset, \quad \forall x \in V \cap dom(g), \tag{1.13}$$

then $x^\star$ is a local minimizer of $g - h$.

Corollary 1.2 *If $x \in int(dom(h))$ verifies*

$$\partial h(x) \in int(\partial g(x)),$$

then x is a local minimizer of $g - h$.

Corollary 1.3 *If $h \in \Gamma_0(X)$ is polyhedral convex then $\partial h(x) \subset \partial g(x)$ is a necessary and sufficient condition for x is a local minimizer of $g - h$.*

Proof 1.3 *This result generalizes the first obtained by C. Michelot in this case where $g, h \in \Gamma_0(X)$ are finite everywhere and h polyhedral convex [Le Thi, 1997, Le Thi and Pham Dinh, 1997].*

For solving a DC optimization problem, it is sometimes easier to solve the dual problem (D) than the primal problem (P). Theorem 1.1 provides transportation by duality of global minimizers. We establish the same duality transportation of local minimizers.

Corollary 1.4 ***(DC duality transportation of local minimizers [Le Thi, 1997, Le Thi and Pham Dinh, 1997])*** *Supposed that $x^\bullet \in dom(\partial h)$ is a local minimizer of $g-h$, let $y^\bullet \in \partial h(x^\bullet)$ and $V_{x^\bullet}$ a neighborhood of $x^\bullet$ such that $g(x)-h(x) \geq g(x^\bullet)-h(x^\bullet), \quad \forall x \in V_{x^\bullet} \cap dom(g)$. If*

$$x^\bullet \in int(dom(g^\star)) \quad and \quad \partial g^\star(y^\bullet) \subset V_{x^\bullet}, \tag{1.14}$$

then $y^\bullet$ is a local minimizer of $h^\star - g^\star$.

Proof 1.4 *It is implied immediately from the Proposition 1.1 by restricting f in the interval $V_{x^\bullet} \cap dom(g)$.*

Remark 1.4 *Obviously, by duality, all the results in this section are transposed to the dual problem D. For example: if y is a local minimizer of $h^\star - g^\star$, then $\partial g^\star(y) \subset \partial h^\star(y)$.*

1.2.3 DCA

This is a new method based on subgradient optimality and duality in DC optimization (non differential). This approach is completely different from classical subgradient methods in convex optimization. In DCA, the construction algorithm seeks to exploit the structure of the DC problem. It requires, first, to have a DC representation of the function to minimize, i.e. $f = g - h$ (g, h convex), because all transactions work only with the convex components. The sequence of descent directions is obtained by computing a sequence of subgradients not directly from the function f, but from convex components of primal and dual problems.

Principle of DCA

The construction of DCA, discovered by Pham Dinh Tao (1986), is based on characterization of local solutions in DC optimization of primal (P) and dual (D) problems.

$$\alpha = \inf\{g(x) - h(x) : x \in X\} \quad (P),$$

$$\alpha = \inf\{h^\star(y) - g^\star(y) : y \in Y\} \quad (D).$$

The DCA consists of constructing two sequences $\{x^k\}$ and $\{y^k\}$. The first sequence is a candidate to be a solution of the primal problem and the second of the dual problem. These two sequences are related by duality and verify the following properties:

- the sequences $\{g(x^k) - h(x^k)\}$ and $\{h^\star(y^k) - g^\star(y^k)\}$ are decreasing,
- and if $(g-h)(x^{k+1}) = (g-h)(x^k)$ then the algorithm stops at $(k+1)^{th}$ iteration and the point x^k (resp. y^k) is a critical point of $g-h$ (resp. $h^\star - g^\star$),
- otherwise every limit point $x^\bullet$ of $\{x^k\}$ (resp. $y^\bullet$ of $\{y^k\}$) is a critical point of $g-h$ (resp. $h^\star - g^\star$).

The algorithm ultimately seeks a couple $(x^\bullet, y^\bullet) \in X \times Y$ such that $x^\bullet \in \partial g^\star(y^\bullet)$ and $y^\bullet \in \partial h(x^\bullet)$.

Schema of simplified DCA

The main idea of the implementation of the algorithm (simple form) is to construct a sequence $\{x^k\}$, verify at each iteration $\partial g(x^k) \cap \partial h(x^{k-1}) \neq \emptyset$, convergence to a critical point $x^\bullet(\partial h(x^\bullet) \cap \partial g(x^\bullet) \neq \emptyset)$ and symmetrically, similar way by duality, a sequence $\{y^k\}$ such that $\partial g^\star(y^{k-1}) \cap \partial h^\star(y^k) \neq \emptyset$ convergence to a critical point.
They are constructed as follows. (see Algorithm 1)

Algorithm 1 DCA

Step 0. Choose an initial point x^0.
Step 1. For each k, x^k is known, computing $y^k \in \partial h(x^k)$.
Step 2. Finding $x^{k+1} \in \partial g^\star(y^k)$.
Step 3. If stopping test is verified **STOP**; otherwise $k \leftarrow k+1$.

This description, with the help of iteration diagrams of fixed points of multi-applications ∂h and $\partial g^\star$, thus appears to be very simple.

Existence of generated sequences

The DCA algorithm is well defined if we can actually build the two sequences $\{x^k\}$ and $\{y^k\}$ as above from an arbitrary initial point x^0.

- By construction, if $x^0 \in \mathrm{dom}(\partial h)$, then $y^0 \in \partial h(x^0)$ is well defined.
- For $k \geq 1$, y^k is well defined if and only if x^k is defined and contained in $\mathrm{dom}(\partial h)$; consequently, x^k and y^k are well defined if and only if $\partial g^\star(y^{k+1}) \cap \mathrm{dom}(\partial h)$ is non empty, which implies that $y^{k+1} \in \mathrm{dom}(\partial g^\star)$.

Lemma 1.1 *[Le Thi and Pham Dinh, 1997] The sequences $\{x^k\}$, $\{y^k\}$ in DCA are well defined if and only if*

$$dom(\partial g) \subset dom(\partial h), \quad \text{and} \quad dom(\partial h^\star) \subset dom(\partial g^\star).$$

The convergence of the algorithm is ensured by the following results [Le Thi and Pham Dinh, 1997]:

Let ρ_i and $\rho_i^\star, (i = 1, 2)$ be positive real numbers such that $0 \leq \rho_i < \rho(f_i)$ (resp. $0 \leq \rho_i^\star < \rho_i^\star(f_i^\star)$) where $\rho_i = 0$ (resp. $\rho_i^\star = 0$) if $\rho(f_i) = 0$ (resp. $\rho(f_i^\star) = 0$) and ρ_i (resp. $\rho_i^\star$) can take the value $\rho(f_i)$ (resp. $\rho(f_i^\star)$) if this upper bound is reached. We consider $f_1 = g, f_2 = h$.

Theorem 1.4 *If the sequences $\{x^k\}$ and $\{y^k\}$ are well defined, then we have:*

(i)

$$(g-h)(x^{k+1}) \leq (h^\star - g^\star)(y^k) - \frac{\rho_h}{2}\|dx^k\|^2 \leq (g-h)(x^k) - \frac{\rho_1+\rho_2}{2}\|dx^k\|^2$$

(ii)

$$(h^\star - g^\star)(y^{k+1}) \leq (g-h)(x^{k+1}) - \frac{\rho_1^\star}{2}\|dy^k\|^2 \leq (h^\star - g^\star)(y^k) - \frac{\rho_1^\star+\rho_2^\star}{2}\|dy^k\|^2$$

where $dx^k = x^{k+1} - x^k$

Corollary 1.5 *([Le Thi and Pham Dinh, 1997])* ***(Convergence)***

1.
$$\begin{aligned}(g-h)(x^{k+1}) &\leq (h^\star - g^\star)(y^k) - \tfrac{\rho_2}{2}\|dx^k\|^2 \\ &\leq (g-h)(x^k) - [\tfrac{\rho_2}{2}\|dx^{k-1}\|^2 + \tfrac{\rho_1^\star}{2}\|dy^k\|^2]\end{aligned}$$

2.
$$\begin{aligned}(g-h)(x^{k+1}) &\leq (h^\star - g^\star)(y^k) - \tfrac{\rho_2^\star}{2}\|dx^k\|^2 \\ &\leq (g-h)(x^k) - [\tfrac{\rho_2^\star}{2}\|dx^{k-1}\|^2 + \tfrac{\rho_1^\star}{2}\|dy^k\|^2]\end{aligned}$$

3.
$$\begin{aligned}(h^\star - g^\star)(y^{k+1}) &\leq (g-h)(x^{k+1}) - \tfrac{\rho_1^\star}{2}\|dy^k\|^2 \\ &\leq (h^\star - g^\star)(y^k) - [\tfrac{\rho_1^\star}{2}\|dy^k\|^2 + \tfrac{\rho_2^\star}{2}\|dx^k\|^2]\end{aligned}$$

4.
$$\begin{aligned}(h^\star - g^\star)(y^{k+1}) &\leq (g-h)(x^{k+1}) - \tfrac{\rho_1}{2}\|dy^{k+1}\|^2 \\ &\leq (h^\star - g^\star)(y^k) - [\tfrac{\rho_1}{2}\|dx^{k+1}\|^2 + \tfrac{\rho_2}{2}\|dx^k\|^2]\end{aligned}$$

Corollary 1.6 *([Le Thi and Pham Dinh, 1997]) If the equalities take place, we have:*

1. $(g-h)(x^{k+1}) = (h^\star - g^\star)(y^k) \Longleftrightarrow y^k \in \partial h(x^{k+1})$
2. $(g-h)(x^{k+1}) = (g-h)(x^k) \Longleftrightarrow x^k \in \partial g^\star(y^k), \quad y^k \in \partial h(x^{k+1})$
3. $(h^\star - g^\star)(y^k) = (g-h)(x^k) \Longleftrightarrow x^k \in \partial g^\star(y^k)$
4. $(h^\star - g^\star)(y^{k+1}) = (h^\star - g^\star)(y^k) \Longleftrightarrow y^k \in \partial h(x^{k+1}), \quad x^{k+1} \in \partial g^\star(y^{k+1})$

In general, the qualities (robustness, stability, convergence rate, good local solutions) of DCA depend on DC decomposition of the objective function $f = g - h$. Theorem 1.4 shows that the strong convexity of convex components in primal and dual problems can affect DCA. To make convex components g and h strongly convex, we can usually apply the following operation

$$f = g - h = \left(g + \frac{\lambda}{2}\|.\|^2\right) - \left(h + \frac{\lambda}{2}\|.\|^2\right).$$

In this case, the convex components in the dual problem will be continuously differentiable.

Computation of subgradients

The description of DCA iteration schemes uses fixed points of multi-applications ∂h and $\partial g^\star$ (∂g and $\partial h^\star$), this can be expressed as follows:

$$\begin{array}{lcl} x^k & \leftarrow & y^k \in \partial h(x^k) \\ & \swarrow & \\ x^{k+1} \in \partial g^\star(y^k) & \leftarrow & y^{k+1} \in \partial h(x^{k+1}) \\ (y^k \in \partial g(x^{k+1})) & & (x^{k+1} \in \partial h^\star(y^{k+1})) \end{array} \tag{1.15}$$

We see a perfect symmetry of the action of two sequences $\{x^k\}$ and $\{y^k\}$ on the dual DC optimization.

The calculation of the subgradient of the function h at a point x^k is usually easy, in many practical problems we know the explicit expression of ∂h. In contrast, the calculation of a subgradient in the conjugate of the convex function g at point y^k usually requires solving the convex program,

$$\partial g^\star(y^k) = argmin\{g(x) - \langle y^k, x\rangle : x \in X\}. \tag{1.16}$$

Indeed, recall that the explicit expression of the conjugate of a given function, in practice, is not known. From (1.16), note that the computation of x^{k+1} is equivalent to minimizing a convex function derived from DC function $f = g - h$, by approximating the concave component $-h$ by its affine minorization at the point x^k, i.e.

$$x^{k+1} \in \partial g^\star(y^k): \quad x^{k+1} \in argmin\{g(x) - \Big[\langle y^k, x - x^k\rangle + h(x^k)\Big] : x \in X\}.$$

And similarly, by duality:

$$y^{k+1} \in \partial h(x^{k+1}): \quad y^{k+1} \in argmin\{h^\star(y) - \Big[\langle x^{k+1}, y - y^k\rangle + g^\star(y^k)\Big] : y \in Y\}.$$

Polyhedral DC optimization

Polyhedral DC optimization occurs when one convex component g or h is polyhedral convex. Like the polyhedral convex optimization problems, this class of problems of DC optimization is frequently encountered in practice and has interesting properties. We will see that the description of DCA is particularly simple ([Le Thi, 1994, 1997, Le Thi and Pham Dinh, 1997]).

Consider a DC program

$$\inf\{g(x) - h(x) : x \in X\} \quad (P).$$

When the convex component h is polyhedral, i.e.

$$h(x) = \max_{x \in X}\{\langle a^i, x\rangle - b^i : i = 1, ..., m\},$$

then the calculation of the subgradients $y^k = \partial h(x^k)$ is explicit. It is clear that limiting (naturally) the choice of subgradients or gradients of affine minorization function h, i.e. $\{y^k\} \in \{a^i : i = 1, ..., m\}$, which is a finite set, following the iteration $\{y^k\}$ is finite ($k \leq m$). Indeed, the sequence $\{(h^\star - g^\star)(y^k)\}$ is, by construction of DCA, decreasing and the choices of iterations y^k are finite. Similarly, by duality the sequence $\{x^k\}$ is finite and $\{(g-h)(x^k)\}$ is decreasing.

Theorem 1.5 ***(Finite convergence)***

- *the sequences* $\{g(x^k) - h(x^k)\}$ *and* $\{h^\star(y^k) - g^\star(y^k)\}$ *are decreasing,*
- *when* $(g - h)(x^{k+1}) = (g - h)(x^k)$ *then the algorithm stops at* $(k + 1)^{th}$ *iteration and a point* x^k *(resp.* y^k*) is a critical point of* $g - h$ *(resp.* $h^\star - g^\star$*).*

Note that if the convex component g is polyhedral, the conjugate function g^* is polyhedral too and writing the dual problem, we found the same results as above.

DCA interpretations

At each iteration of DCA, we replace the second component h in primal DC program by its affine minorization $h_k(x) = h(x^k) + \langle x - x^k, y^k \rangle$ in a neighborhood of x^k to obtain the convex program following

$$\inf\{\overline{f}_k = g(x) - h_k(x) : x \in \mathbb{R}^n\} \tag{1.17}$$

whose set of optimal solutions is $\partial g^\star(y^k)$.

Similarly, the second DC component $g^\star$ in dual DC program (1.6) is replaced by its affine minorization $(g^\star)_k(y) = g^\star(y^k) + \langle y - y^k, x^{k+1} \rangle$ in a neighborhood of y^k to give birth to the convex program

$$\inf\{h^\star(y) - (g^\star)_k(y) : y \in \mathbb{R}^n\} \tag{1.18}$$

which $\partial h(x^{k+1})$ is the set of optimal solutions. It should be noted that DCA works with DC components g and h and not with the function f itself. Each DC decomposition of f gives rise to a DCA.

Such as $\overline{f}_k$ is a convex function, the minimizer x^{k+1} is defined by $0 \in \partial \overline{f}_k(x^{k+1})$ and the minorization of f by $\overline{f}_k$ ensures the decreasing of the sequence $\{f(x^k)\}$. Indeed, as h_k is a affine minorization function of h at x_k, $\overline{f}_k$ is a minorization convex function of f,

$$f(x) \leq \overline{f}_k(x), \quad \forall x \in X,$$

which coincides at x^k with f,

$$f(x^k) = \overline{f}(x^k).$$

Therefore determining the iteration x^{k+1} as the minimum convex program (1.17), the decreasing of the sequence of iterations is endured,

$$f(x^{k+1}) \leq f(x^k).$$

If at the iteration $k+1$, $f(x^{k+1}) = f(x^k)$ then x^{k+1} is a critical point of f ($0 \in \partial \overline{f}_k(x^{k+1}) \Longrightarrow 0 \in \partial f(x^{k+1})$).

Remark 1.5 *If $\overline{f}_k$ is strictly convex then there exists a unique minimizer x^{k+1}.*

Comment: It is important to note that the function f is replaced by function $\overline{f}_k$ in overall domain of f,

$$\overline{f}_k(x) = g(x) - (\langle y^k, x - x^k \rangle + h(x^k)) \quad \text{with} \quad y^k \in \partial h(x^k), \forall x \in X$$

which, considered locally in the neighborhood of x^k, is a first order approximation of f and globally on $\mathbb{R}^n$. It is noteworthy that $\overline{f}_k$ is not defined narrowly from local information of f at neighborhood of x^k (i.e. $f(x^k), \partial f(x^k), ...$) but incorporates all of the first convex components of f in its definition, i.e. $\overline{f}_k = g - h_k = f - (h + h_k)$. In other words, $\overline{f}_k$ is not simply a local approximation of f in a neighborhood of x^k, but should rather be described as "convexification majorant" of f globally related to DC function by the first convex component defined on $\mathbb{R}^n$. Therefore, no displacement of the x^k to x^{k+1} is determined from f globally defined for all $x \in \mathbb{R}^n$. DCA can not be simply regarded as a local approximation method or a local descent method in classic, the global characteristic is the "convexification majorant". Thus, unlike conventional local approaches (deterministic or heuristic), DCA operates simultaneously local and global properties of the function to be minimized during the iterative process and in practice converges to a good solution local

and sometimes global.

For a comprehensive study of DC programming and DCA, refer to [Le Thi, 1994]-[Le Thi et al., 2002] and [Pham Dinh, 1988]-[Pham Dinh and Le Thi, 1998] and the reference therein. The treatment of a nonconvex problem by DC approach and DCA should have two tasks: looking for an appropriate DC decomposition and looking for a good starting point.

For a DC program given, the issue of finding a good DC decomposition remains open, in practice, we look for a DC decomposition to adapt with the structure of the DC program for which the studied sequences $\{x^k\}$ and $\{y^k\}$ are easy to calculate. If the calculation is explicit then the corresponding DCA is less expensive time and it is able to support very large dimensions.

1.3 DCA and DCA B&B method for solving MILP0-1

1.3.1 DCA for solving MILP0-1

Restatement

Consider the mixed 0-1 integer linear programming in the form:

$$(MILP01) \qquad \begin{cases} \min f(x,y) = c^T x + d^T y \\ s.t. \\ \quad Ax + By \leq b, \\ \quad x \in \{0,1\}^n, \\ \quad y \in \mathbb{R}^p_+. \end{cases} \tag{1.19}$$

where A and B are two matrices with the dimension $(m \times n)$ and $(m \times p)$ (resp.), m - the number of constraints, n - the number of binary variables, p - the number of continuous variables, $(c,d) \in \mathbb{R}^n \times \mathbb{R}^p$ - the cost vectors and $b \in \mathbb{R}^m$.

We define two sets S (the set of feasible solutions) and K (the set of feasible solutions of the standard linear relaxation) in the following:

$$S := \{(x,y) \in \{0,1\}^n \times \mathbb{R}^p_+ \ : \ Ax + By \leq b\},$$
$$K := \{(x,y) \in [0,1]^n \times \mathbb{R}^p_+ \ : \ Ax + By \leq b\}.$$

Consider the function p defined by:

$$p(x,y) \equiv p(x) := \sum_{j=1}^{n} x_j(1 - x_j).$$

It is clear that:

- p is concave on $\mathbb{R}^n \times \mathbb{R}^p$,
- $z \in K$ is a feasible point of (MILP01) if and only if $p(z) = 0$,
- p is non negative, finite on K and we have:

$$S := \{(x,y) \in \{0,1\}^n \times \mathbb{R}^p_+ : Ax + By \leq b\}$$
$$\equiv \{(x,y) \in K : \ p(x) = 0\}$$

$$\equiv \{(x,y) \in K : \ p(x) \leq 0\}.$$

Therefore, (MILP01) can be written in the form of a nonconvex continuous problem defined as follows:

$$(NCP1) \quad \min\{c^T x + d^T y : (x,y) \in K, p(x) \leq 0\}. \tag{1.20}$$

Theorem 1.6 *Let K be a nonempty bounded polyhedral convex set on $\mathbb{R}^n$. Let f be a finite concave function on K and p be a finite nonnegative concave function on K. Then there exists $t_0 \geq 0$ such that for $t \geq t_0$ the following problems have the same optimal value and the same solution set:*

$$\alpha(t) = \inf\{f(x) + tp(x) : x \in K\},$$

$$\alpha = \inf\{f(x) : x \in K, p(x) \leq 0\}.$$

Furthermore, if the vertex set of K, denoted by $V(K)$ is contained in $\{x \in K, p(x) \leq 0\}$, then $t_0 = 0$, otherwise $t_0 = \min\{\frac{f(x)-\alpha(0)}{\xi}\}$, where $\xi := \min\{p(x) : x \in V(K), p(x) > 0\} > 0$.

Proof 1.5 *See H.A. Le Thi, T. Pham Dinh et M. Le Dung [Le Thi et al., 1999].*

From Theorem 1.6, there exists a number t_0 such that $\forall t \geq t_0$, (NCP1) is equivalent to the concave minimization problem in the following:

$$(NCP2) \quad \min\{c^T x + d^T y + tp(x) : (x,y) \in K\}.$$

Note that, the the non-convexity constraints of problem $(NCP1)$ is taken into account in the objective function of problem $(NCP2)$. We can reformulate $(NCP2)$ as a DC program, then DCA can be used to solve it.

DCA applied to the penalized problem MILP01

The problem $(NCP2)$ can be rewritten as

$$\min\{\chi_K(z) + c^T x + d^T y + tp(x) \ : \ z = (x,y) \in \mathbb{R}^n \times \mathbb{R}^p\}, \tag{1.21}$$

where

$$\chi_K(z) := \begin{cases} 0 & \text{if } z \in K, \\ +\infty & \text{otherwise} \end{cases} \tag{1.22}$$

is the indicator function of K.

We note $g(z) := \chi_K(z)$ and

$$h(z) := -c^T x - d^T y + t(-p)(z) = -c^T x - d^T y + t\sum_{j=1}^{n} x_j(x_j - 1). \tag{1.23}$$

Since K is convex, $g := \chi_K$ is convex [Rockafellar, 1970]. Also, h is convex and p is concave. Therefore, the problem $(NCP2)$ is equivalent to the following DC program:

$$\min\{g(z) - h(z) \ : \ z = (x,y) \in \mathbb{R}^n \times \mathbb{R}^p\}. \tag{1.24}$$

According to the general diagram of DCA, we must compute the two suites $\{u^k\}$ and $\{z^k\}$ such that

$$u^k \in \partial h(z^k); \quad z^{k+1} \in \partial g^*(u^k).$$

Let $z = (x, y)$ be an arbitrary point of $\mathbb{R}^n \times \mathbb{R}^p$; from the definition (1.23) of function h, a vector gradient in $u = (\sigma, \varsigma) \in \partial h(x, y)$ is chosen by:

$$\sigma_i = -c_i + 2tx_i - t \text{ and } \varsigma_j = -d_j \quad (i = 1, \ldots, n; j = 1, \ldots, p). \tag{1.25}$$

Then the computation of z^{k+1} is equivalent to solving the following problem:

$$\min\{g(z) - \langle z, u\rangle : z \in \mathbb{R}^n \times \mathbb{R}^p\}. \tag{1.26}$$

It can be rewritten as

$$\min\{-\langle z, u\rangle : z \in K\}. \tag{1.27}$$

The application of DCA to problem (1.24) can be written as follows:

Algorithm 1.1 *Application of DCA to problem (1.24)*

Step 1. *Let $z^0 = (x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^p$ be an initial point. Set $k \leftarrow 0$;*

Step 2. *Calculate $u^k = (\sigma^k, \varsigma^k) \in \partial h(x^k, y^k)$ via (1.25).*

Step 3. *Calculate*

$$z^{k+1} = (x^{k+1}, y^{k+1}) \in \arg\min\{-\langle(\sigma^k, \varsigma^k), (x, y)\rangle : (x, y) \in K\}. \tag{1.28}$$

Step 4.

If the stop criterion is verified, STOP;

Else return to Step 2;

Thanks to the theorem on the convergence of DCA for polyhedral programming [Le Thi and Pham Dinh, 2001, 2005, Le Thi et al., 2007] we have directly following properties of Algorithm 1.1.

Theorem 1.7 *Convergence properties of Algorithm 1.1*

i) Algorithm 1.1 generates a sequence $\{z^k = (x^k, y^k)\}$ contained in the vertex set $V(K)$ of K such that sequence $\{g(x^k, y^k) - h(x^k, y^k)\}$ is decreasing.

ii) If at iteration r the point (x^r, y^r) satisfies $x^r \in \{0, 1\}^n$, then (x^k, y^k) satisfies $x^k \in \{0, 1\}^n$, for all $k \geq r$.

iii) The sequence $\{(x^k, y^k)\}$ converges to a solution $(x^\star, y^\star) \in V(K)$ after a finite number of iterations and $(x^\star, y^\star)$ is a critical point of the problem (1.24). Moreover, if $x_i^\star \neq 0.5, \quad \forall i = 1, \ldots, n$, then $(x^\star, y^\star)$ is a local solution of problem(1.24).

Proof 1.6 *See, for example, H.A. Le Thi* et al. *[Le Thi et al., 2007].*

1.3.2 B&B for MILP01

This section discusses a particular case of B&B for mixed 0-1 integer linear problem (MILP01). Consider the following problem:

$$(MILP01) \quad \begin{cases} \min f(x, y) = c^T x + d^T y \\ s.t. \\ \quad Ax + By \leq b, \\ \quad x \in \{0, 1\}^n, \\ \quad y \in \mathbb{R}_+^p. \end{cases} \tag{1.29}$$

Let us set $z = (x, y)$ and

- $S := \{z = (x, y) : Ax + By \leq b, x \in \{0,1\}^n, y \in \mathbb{R}^p_+\}$ be the feasible set of (MILP01),
- R_0 be the relax domain of S:

$$R_0 = \{z = (x, y) : Ax + By \leq b, x \in [0,1]^n, y \in \mathbb{R}^p_+\},$$

- $\mathcal{R}$ be the set of all R_i that can be divided,
- β_k (resp. γ_k) be the lower bound (resp. the upper bound) of optimal value at iteration k.
- z^k be the best known solution at iteration k (as we do not yet know whether it is a feasible point, $\gamma_k = \infty$),
- and $\overline{z}$ be the optimal solution of relaxed problem of (MILP0-1).

Principle: We determined, at each iteration k, a lower bound β_k and an upper bound γ_k of the optimal value of problem (MILP01), such that $\gamma_k - \beta_k \to 0$. We stop when $\gamma_k - \beta_k \leq \varepsilon$. Specifically:

- the lower bound β_k is calculated by

$$\beta_k = \min\{\beta(R_i) : R_i \in \mathcal{R}\} = \min_{R_i \in \mathcal{R}}\{f(z) : z \in R_i\}.$$

- the best upper bound at iteration k is $\gamma_k = \min\{f(z_i) : \forall z_i \text{ known in } S\}$.
- at each iteration k, we consider the set $\mathcal{R} = \{R_i \in \mathcal{R} : \beta(R_i) < \gamma_k\}$ and we remove all the $R_i \in \mathcal{R}$ for which $\beta(R_i) \geq \gamma_k$ (because we are sure that R_i does not contain an optimal solution if $\beta(R_i) > \gamma_k$; in case where $\beta(R_i) = \gamma_k$ we can not determine on R_i a solution better than z^k).

Algorithm 1.2 ***(B&B)***

Step 0 (Initialization)

0.1. Calculate β_0 and $\overline{z}$ by solving the problem ($MILP01'$) relaxed problem of (MILP01).

- *If $\overline{z} \in S$ then STOP, $\overline{z}$ is an optimal solution of (MILP01).*
- *Otherwise, set $\mathcal{R} := \{R_0\}$, $\gamma_0 = +\infty$, $k := 1$.*

0.2. Go to the procedure of choice (Step 1).

Step 1 (Procedure of choice)

1.2. Choose a subset $R_k \in \mathcal{R}$ such that

$$\beta_k = \min\{\beta(R_i) : R_i \in \mathcal{R}\}.$$

1.2. Go to the branch procedure (Step 2).

Step 2 (Branch procedure)

2.1. Let r be an index for which $\overline{x}^k_r$ is not binary. Separate R_k into two subsets:

$$R_{k1} = \{z \in R_k : x_r = 0\}; \qquad R_{k2} = \{z \in R_k : x_r = 1\}.$$

2.2. Go to the bounds procedure (Step 3).

Step 3 (Bounds procedure)

3.1. Calculate β_{k1} and β_{k2} by solving the two problems ($MILP01'_{k1}$) and ($MILP01'_{k2}$).

3.2. If $\overline{z}^{k1} \in S$ ($\overline{z}^{k2} \in S$ resp.), and $f(\overline{z}^{k1}) < \gamma_k$ (resp. $f(\overline{z}^{k2}) < \gamma_k$) then update γ_k

$$\gamma_k := f(\overline{z}^{k1}); z^k := \overline{z}^{k1}; \qquad \big(resp.\ \gamma_k := f(\overline{z}^{k2}); z^k := \overline{z}^{k2}\big).$$

Step 4 (Optimal test)

4.1. $\mathcal{R} := \mathcal{R} \cup \{R_{ki} : f(\overline{z}^{ki}) < \gamma_k, i = 1, 2\} \setminus \{R_k\}$.

4.2. If $\mathcal{R} = \emptyset$ then STOP, z^k is an optimal solution, otherwise $k := k + 1$ and go to the procedure of choice (Step 1).

For the procedures selecting a subset R_k to separate, we can also use another procedure of choice called "exploration depth-first" of tree (backtracking): select R_k the most recently created. This procedure has two advantages and one disadvantage:

- It allows to quickly obtain a feasible solution,
- It allows to minimize the transfers between main memory and the peripheral memory.

We may create a combination of both procedures displayed: using a backtracking strategy to "go down" in the tree, but when we found a sterile subset we choose the subset maximum score non-sterile.

For procedures separation, we have certain rules to be applied to determine the index r:

- choose r for which one of two programs ($MILP01'_{k1}$) and ($MILP01'_{k2}$) has no feasible solution (false separation);
- choose r for which $\overline{x}_r$ ($MILP01'_k$) is the fractional possible (its value is as close as possible to 0.5).

1.3.3 B&B combined with DCA

In scheme B&B (shown in 1.3.2), the problem of finding a good upper bound plays an important role for the efficiency of B&B algorithm. Several studies in the literature showed that the value obtained by DCA is often very close/coincides with the optimal value (especially when we restart DCA from a good starting point). We combine B&B and DCA in order to find a good starting point for DCA and prove/find the global solution.

Recall that MILP01 is transformed into a DC programming. We apply DCA to (NCP2) (see Section 1.3.1) for determining the upper bound of (MILP01).

Using the same notation as in Section 1.3.2, the algorithm B&B combined with DCA can be described as follows:

Algorithm 1.3 ***(BBDCA)***

Step 0 (Initialization)

0.1. Calculate β_0 and $\overline{z}$ by solving the problem ($MILP01'$) relaxed of (MILP01).

- *If $\overline{z} \in S$ then STOP, $\overline{z}$ is an optimal solution of (MILP01).*
- *Else, set $\mathcal{R} := \{R_0\}$ and $k := 0$.*

0.2. Apply DCA to problem (NCP2) from $\overline{z}$ to obtain z^k_{DCA}.

- *If $z^k_{DCA} \in S$, then $\gamma_k := f(z^k_{DCA})$.*
- *Else, $\gamma_k := +\infty$.*

0.3. Set $k := 1$ and go to the procedure of choice (Step 1).

Step 1 (Procedure of choice)

1.2. Choose a subset $R_k \in \mathcal{R}$ such that

$$\beta_k = \min\{\beta(R_i) : R_i \in \mathcal{R}\}.$$

1.2. Go to the separation (branch) procedure (Step 2).

Step 2 (Procedure of separation (branch))

2.1. Let r for which an index $\overline{x}^k_r$ is not binary. Separate R_k to two subsets

$$R_{k1} = \{z \in R_k : x_r = 0\}; \qquad R_{k2} = \{z \in R_k : x_r = 1\}.$$

2.2. Go to the assessment procedure (Step 3).

Step 3 (Procedure of assessment (calculated bound))

3.1. Calculate β_{k1} and β_{k2} by solving two problems $(MILP01'_{k1})$ and $(MILP01'_{k2})$.

3.2. Apply DCA to (NCP2) from $\overline{z}^{k1}$ and $\overline{z}^{k2}$ to obtain z^{k1}_{DCA} and z^{k2}_{DCA}.

3.3. If $z^{k1}_{DCA} \in S$ (resp. $z^{k2}_{DCA} \in S$) and $f(z^{k1}_{DCA}) < \gamma_k$ (resp. $f(z^{k2}_{DCA}) < \gamma_k$) then update γ_k

$$\gamma_k := f(z^{k1}_{DCA}); z^k := z^{k1}_{DCA}; \qquad (\textit{resp. } \gamma_k := f(z^{k2}_{DCA}); z^k := z^{k2}_{DCA}).$$

3.4. Go to Step 4.

Step 4 (Test of optimal conditions)

4.1. $\mathcal{R} := \mathcal{R} \cup \{R_{ki} : f(\overline{z}^{ki}) < \gamma_k, i = 1, 2\} \setminus \{R_k\}$.

4.2. If $\mathcal{R} = \emptyset$ then STOP, z^k is a optimal solution. Else $k := k+1$ and go to procedure of choice (Step 1).

Note that, in the algorithm 1.3, step 3.2 is not always performed at each iteration.

The question *when DCA is restarted* (step 3.2 is executed) is interesting from numerical points of view and it will be studied. As in several DC programs, DCA provides a global solution to (NCP2) (and so to (MILP)) from a *good* starting point. Such a point can be found efficiently while computing lower bounds. We will see in the computational experiments that DCA provides an integer solution after several iterations of the Branch and Bound algorithm. Nevertheless we must continue the Branch and Bound process to ameliorate the lower bound until it is close to the best upper bound. In fact, the Branch and Bound algorithm is introduced to find a good starting point for DCA and check the globality of DCA.

Finding a good starting point for DCA

From Theorem 1.7 we see that, starting with a feasible solution to (MILP) DCA provides a better feasible solution, although it works on a continuous feasible set of (NCP2). It is so important to find a good feasible point to (MILP) for restarting DCA.

During the branch and bound procedure we can restart DCA from the best feasible solution to (MILP) that is discovered while computing lower bounds. This is motivated by a similar and efficient way introduced in the combined DCA-Branch and Bound algorithm for nonconvex quadratic programming ([Le Thi and Pham Dinh, 2001]).

On the other hand, for obtaining rapidly a good feasible point to (MILP), we also investigate a fast procedure to compute an optimal solution of the concave quadratic program

$$0 = \min\Big\{\sum_{i=1}^{m} x_i(1-x_i) : (x,y) \in K\Big\}. \tag{1.30}$$

That is the DCA developed in [Le Thi and Pham Dinh, 2001].

In our algorithm, a suitable starting point is computed by choosing one of the two procedures according to the current situation.

When DCA is restarted?

During the branch and bound process, we restart DCA when a feasible solution to (MILP), which improves the best current upper bound, is pointed out. In such a case, the starting point of DCA is the just mentioned feasible solution to (MILP).

On the other hand, DCA is also restarted when the number of the 0-1 components of the binary variables (denoted by $N_{x^{R_k}}$) of the solution (x^{R_k}, y^{R_k}) to the corresponding linear relaxation problem is sufficiently large, namely $N_{x^{R_k}} \geq m/2$. The starting point of DCA, in this case, is the solution of Problem (1.30).

1.4 Proximal Decomposition Method (PDM)

This section provides a brief presentation of Proximal Decomposition Method which is applied to solve convex optimization problems.

1.4.1 Proximal decomposition on the graph of a maximal monotone operator

Definition 1.3 *Let H be a real Hilbert space with inner product $\langle .,. \rangle$. A multifunction $T : H \rightrightarrows H$ is called a* monotone operator *if*

$$\langle x - x', y - y' \rangle \geq 0, \ \forall y \in T(x), \ y' \in T(x'). \tag{1.31}$$

The graph of T is defined by

$$Gr(T) := \{(x,y) \in X \times X | \ y \in T(x)\}. \tag{1.32}$$

Definition 1.4 *An operator T is called* maximal monotone *if T is monotone and there dose not exist a monotone operator T' such that $Gr(T) \subsetneq Gr(T')$.*

Since T is maximal, its graph is not properly contained in the graph of any other monotone operator.

An operator T is called *strongly monotone* if there exists $\rho > 0$ such that:

$$\langle x - x', y - y' \rangle \geq \rho ||x - x'||^2, \quad \forall x, x' \in X \text{ and } \forall y \in T(x),\ \forall y' \in Tx', \tag{1.33}$$

and value $\rho_0 = \max\{\rho > 0 \text{ satisfying } (1.33)\}$ is called modulus of T.

An operator T is called *Lipschitz with constant* L if

$$||y - y'|| \leq L||x - x'||, \quad \forall x, x' \in X \text{ and } \forall y \in T(x),\ \forall y' \in Tx'.$$

Theorem 1.8 *If T is maximal monotone, then $(I+\lambda T)^{-1}$ is unique and $dom(I+\lambda T)^{-1} = X$ for all $\lambda > 0$.*

It means that, for each $y \in X$ and $\lambda > 0$, there is unique $x \in X$ such that $y \in (I + \lambda T)(x)$ and the operator $P = (I + \lambda T)^{-1}$ is *nonexpansive,* i.e.,

$$||P(x) - P(x')|| \leq ||x - x'||. \tag{1.34}$$

Note that $P(x) = x$ if and only if $0 \in \lambda T(x)$ and P is called *Proximal mapping* associated with λT.

The *proximal point algorithm* which aims to find x^* such that $0 \in T(x^*)$ constructs a sequence $x^{(k)}$ from an arbitrary initial point x^0 by the equation

$$x^{(k+1)} = (I + \lambda_k T)^{-1}(x^{(k)}) \tag{1.35}$$

where $\{\lambda_k\}$ is a sequence of positive real numbers.

Proposition 1.7 *Let T be a maximal monotone operator on X. Then the resolvent $(I + T)^{-1}$ is firmly nonexpansive, i.e.,*

$$||(I+T)^{-1}(x)-(I+T)^{-1}(x')||^2 \leq \langle x-x', (I+T)^{-1}(x)-(I+T)^{-1}(x')\rangle \quad \forall x, x' \in X. \tag{1.36}$$

Moreover, if T is strongly monotone with modulus ρ, then

$$||(I+T)^{-1}(x)-(I+T)^{-1}(x')||^2 \leq \frac{1}{1+\rho}\langle x-x', (I+T)^{-1}(x)-(I+T)^{-1}(x')\rangle \quad \forall x, x' \in X. \tag{1.37}$$

The decomposition on the graph of a maximal monotone operator was first introduced in (see [Mahey et al., 1995]), it is presented as follows.

Proposition 1.8 *Given a maximal monotone operator T and a vector $(x, y) \in X \times X$, there exists a unique pair $(u, v) \in X \times X$ such that: $u + v = x + y$ and $(u, v) \in Gr(T)$.*

From Proposition 1.8, we see that the proximal decomposition of (x, y) on the graph of T is unique, and calculated by:

$$\begin{aligned} u &= (I + T)^{-1}(x + y), \\ v &= (I + T^{-1})^{-1}(x + y) \\ &= x + y - u. \end{aligned}$$

As a result in [Mahey et al., 1995], the proximal decomposition on the graph of a maximal monotone operator is nonexpansive, when restricted to a product of orthogonal subspaces of $X \times X$.

Theorem 1.9 *Let A and B be two complementary orthogonal subspaces of $X \times X$. The mapping $\mathcal{F}$ which, to any $(x, y) \in A \times B$, associates its proximal decomposition (u, v) on the graph of T, is nonexpansive. Moreover, if T is strongly monotone with modulus ρ and Lipschitz with constant L, then $\mathcal{F}$ is a contraction mapping.*

1.4.2 Proximal decomposition method

Let T be a maximal monotone operator on X, A be a subspace and B be its orthogonal subspace. We revisit now the problem (P) which was analyzed in [Mahey et al., 1995], [Spingarn, 1983]. The problem consists in finding

$$(x, y) \in X \times X \text{ such that } (x, y) \in A \times B \cap Gr(T). \tag{1.38}$$

The algorithm alternates a proximal decomposition on the graph of T by a projection on $A \times B$. The algorithm can be summarized as Algorithm 2-PDG.

Algorithm 2 PDG

Initialization: $(x_0, y_0) \in A \times B,\ k = 0$
Iteration k
  If $(x_k, y_k) \in Gr(T)$ **then STOP**
  Else
    $z_k = x_k + y_k$
    Proximal step
      $u_k = (I + T)^{-1}(z_k)$
      $v_k = z_k - u_k$
    Projection step
      $x_{k+1} = \text{Proj}_A(u_k)$
      $y_{k+1} = \text{Proj}_B(v_k)$
  $k \leftarrow k + 1$

where $\text{Proj}_C(.)$ is the projection operator on the set C.

Theorem 1.10 *Assume that the set of solutions of (P) is nonempty, then the sequence* $\{(x_k, y_k)\}$ *generated by Algorithm 2-PDG converges to a solution of (P).*

Proof 1.7 *See Theorem 3 in [Mahey et al., 1995].*

The *Proximal point algorithm* can be seen as a particular case of Algorithm 2-PDG when let $A = X$, which implies $B = 0$.

Proximal scaled decomposition

To generalize, we present now a scaled decomposition on the graph of a maximal monotone operator, which was first introduced in [Mahey et al., 1995].

Definition 1.5 *Let* $(x, y) \in X \times X$, T *be a maximal monotone operator and* λ *be a positive number. Then, the scaled proximal decomposition of* (x, y) *on the graph of* T *is the unique* (u, v) *such that:*

$$\begin{aligned} u + \lambda v &= x + \lambda y \\ (u, v) &\in Gr(T) \end{aligned}$$

The existence and unicity of (u, v) can be seen in the following, if $v \in Tu$ then

$$\begin{aligned} u + \lambda v \in u + \lambda T u, \\ \Rightarrow u &= (I + \lambda T)^{-1}(u + \lambda v) \\ &= (I + \lambda T)^{-1}(x + \lambda y), \\ v &= \frac{1}{\lambda}(x + \lambda y - u). \end{aligned}$$

The proximal scaled decomposition algorithm can be viewed as Algorithm 3-SPDG.

Algorithm 3 SPDG

Initialization: $(x_0, y_0) \in A \times B,\ \lambda > 0,\ k = 0$
Iteration k
If $(x_k, y_k) \in Gr(T)$ **then STOP**
Else
$z_k = x_k + \lambda y_k$
Proximal step
$u_k = (I + T)^{-1}(z_k)$
$v_k = \frac{1}{\lambda}(z_k - u_k)$
Projection step
$x_{k+1} = \mathrm{Proj}_A(u_k)$
$y_{k+1} = \mathrm{Proj}_B(v_k)$
$k \leftarrow k + 1$

Theorem 1.11 *When T is strongly monotone with modulus ρ and Lipchitz with constant L, then the convergence of the sequence $\{(x_k, \lambda y_k)\}$ generated by Algorithm 3-(SPDG) is linear with speed ratio $\sqrt{(1 - \frac{2\lambda\rho}{(1+\lambda L)^2})}$.*

Proof 1.8 *See Theorem 4 in [Mahey et al., 1995].*

Solving convex optimization problems by proximal decomposition method

We consider a convex optimization problem,

$$\min\{f(x) |\ x \in C\}, \tag{1.39}$$

where f is l.s.c proper convex function and C is a convex set on $\mathbb{R}^n$.

For solving this problem, it is equivalent to find $x \in C$ such that $0 \in \partial f(x)$ or $0 \in \partial(f + \chi_C)(x)$, where $\chi_C(x)$ is indicator function on C. Recall that f_0 is a l.s.c proper convex function, then ∂f_0 is a maximal monotone operator (see in [Rockafellar, 1976]). We have $(f + \chi_C)(x)$ is a l.s.c proper convex function. Let $T = \partial(f + \chi_C)$, the optimization problem (1.39) can be seen as finding $x \in X$ such that $0 \in T(x)$, where T is a maximal monotone operator. It can be solved by *Proximal point algorithm* . Moreover, when $T = \partial(f + \chi_C)$, the step (1.35) is replaced by (see Moreau [Moreau, 1965])

$$x^{(k+1)} = argmin\{f_{\lambda_k}(x^{(k)}) :\ u \in X\} \tag{1.40}$$

where

$$f_\lambda(x) = f(u) + \chi_C(u) + \frac{1}{2\lambda}||x - u||^2. \tag{1.41}$$

We present now a general case of the aforementioned problem (1.39), when $C = \bigcap_{i=1}^{m} C_i$ where C_i is convex for all $i = 1, \cdots, m$. The problem is rewritten as

$$\min\{f(x) : x \in C = \bigcap_{i=1}^{m} C_i\} \tag{1.42}$$

To solve this problem, we aim to find $0 \in \partial(f + \chi_C)(x)$. Following the theorem 23.8 in [Rockafellar, 1970], it can be rewritten as $0 \in \partial f(x) + \partial\chi_C(x)$ or

$$0 \in \partial f(x) + \sum_{i=1}^{m} \partial\chi_{C_i}(x). \tag{1.43}$$

Proximal decomposition algorithm

Let us set $T_0 = \partial f$ and $T_i = \partial \chi_{C_i}$ for all $i = 1, \cdots, m$. Then we define a maximal monotone operator T as

$$T(x) = T_0(x_0) \times T_1(x_1) \times \cdots \times T_m(x_m) \tag{1.44}$$

where $\mathcal{X} = (x_0, x_1, \cdots, x_m) \in X^{m+1},\ x_i \in X\ \forall i = 1, \cdots, m$.

We define a subspace A and its orthogonal B by

$$A = \{\mathcal{X} = (x_0, x_1, \cdots, x_m) :\ x_0 = x_1 = \cdots = x_m\}, \tag{1.45}$$
$$B = \{\mathcal{Y} = (y_0, y_1, \cdots, y_m) :\ y_0 + y_1 + \cdots + y_m = 0\}. \tag{1.46}$$

Our problem now consists in finding $(\mathcal{X}, \mathcal{Y}) \in A \times B \cap Gr(T)$. We can use Algorithm 2-(PDG) or Algorithm 3-(SPDG) to solve this problem. Note that $(I+T)^{-1}(x) = ((I + T_0)^{-1}(x_0), (I + T_1)^{-1}(x_1), \cdots, (I + T_m)^{-1}(x_m))$ and by [Moreau, 1965], to calculate $(I + T)^{-1}(x)$ we have

$$(I + T_0)^{-1}(x_0) \Leftrightarrow \min\{f(u) + \frac{1}{2}||u - x_0||^2 :\ u \in X\} \tag{1.47}$$
$$(I + T_i)^{-1}(x_i) \Leftrightarrow \min\{\chi_{C_i}(u) + \frac{1}{2}||u - x_i||^2 :\ u \in X\}\ \forall i = 1, \cdots, m, \tag{1.48}$$
$$\Leftrightarrow \text{Proj}_{C_i}(x_i)\ \forall i = 1, 2, \cdots, m. \tag{1.49}$$

The projection step is calculated by

$$\text{Proj}_A(\mathcal{X}) = (\frac{x_0 + \cdots + x_m}{m+1}, \cdots, \frac{x_0 + \cdots + x_m}{m+1}), \tag{1.50}$$
$$\text{Proj}_B(\mathcal{Y}) = (y_0 - \frac{y_0 + \cdots + y_m}{m+1}, \cdots, y_m - \frac{y_0 + \cdots + y_m}{m+1}). \tag{1.51}$$

Proximal copy decomposition algorithm

We replace the problem (1.42) by a new equivalent problem, it is defined as

$$\min\{F(x_1, \cdots, x_m) :\ (x_1, x_2, \cdots, x_m) \in \mathcal{C} = C_1 \times C_2 \times \cdots \times C_m,\ x_1 = x_2 = \cdots = x_m\}, \tag{1.52}$$

where $F(x_1, \cdots, x_m) = \sum_{i=1}^{m} f_i(x_i)$, and $f_i(x) \equiv f(x)\ \forall i = 1, 2, \cdots, m$.

We see that

$$\partial F(x_1, x_2, \cdots, x_m) = \Pi_{i=1}^m \partial f_i(x_i) \tag{1.53}$$
$$\partial \chi_{\mathcal{C}}(x_1, x_2, \cdots, x_m) = \Pi_{i=1}^m \partial \chi_{C_i}(x_i). \tag{1.54}$$

We define subspace A_1 and its orthogonal B_1 as

$$A_1 = \{\mathcal{X}_\infty = (x_1, \cdots, x_m) :\ x_1 = \cdots = x_m\}, \tag{1.55}$$
$$B_1 = \{\mathcal{Y}_\infty = (y_1, \cdots, y_m) :\ y_1 + \cdots + y_m = 0\}, \tag{1.56}$$

then Problem (1.52) becomes to find $(x_1, \cdots, x_m) \in A_1$ such that

$$0 \in \partial(F + \chi_{\mathcal{C}})(x_1, x_2, \cdots, x_m).$$

On the other hand, we have

$$\partial(F + \chi_{\mathcal{C}})(x_1, x_2, \cdots, x_m) = \Pi_{i=1}^m (f_i + \chi_{C_i})(x_i). \tag{1.57}$$

We define a maximal monotone operator $T_1 = \Pi_{i=1}^{m} \partial(f_i + \chi_{C_i})$ then the problem (1.52) is reformulated in the form

$$\text{Finding } (\mathcal{X}_\infty, \mathcal{Y}_\infty) \in A_1 \times B_1 \cap Gr(T_1). \tag{1.58}$$

To solve this problem, in a similar way than with the algorithm above, we calculate:

- Proximal step

$$(I + \partial(f_i + \chi_{C_i}))^{-1}(x_i) \Leftrightarrow \min\{f_i(u) + \chi_{C_i}(u) + \frac{1}{2}||u - x_i||^2 : \ u \in X\} \tag{1.59}$$

$$\Leftrightarrow \min\{f_i(u) + \frac{1}{2}||u - x_i||^2 : \ u \in C_i\} \ \forall i = 1, 2, \cdots, m. \tag{1.60}$$

- Projection step

$$\text{Proj}_{A_1}(\mathcal{X}_\infty) = (\frac{x_1 + \cdots + x_m}{m}, \cdots, \frac{x_1 + \cdots + x_m}{m}), \tag{1.61}$$

$$\text{Proj}_{B_1}(\mathcal{Y}_\infty) = (y_1 - \frac{y_1 + \cdots + y_m}{m}, \cdots, y_m - \frac{y_1 + \cdots + y_m}{m}). \tag{1.62}$$

Applications

One of the most important applications of Proximal decomposition and Proximal copy decomposition (as mentioned above) is to solve convex optimization problems. Especially, when the feasible set of the convex optimization problem is complex, the original problem can be divided into many convex sub-problems associated with simple and smaller feasible sets. They are easier to solve. Then, to solve the original problem, it is replaced by solving the convex sub-problems at each iteration of the algorithm.

1.5 Conclusion

Section 1.2 introduces all the prerequisite notions and definitions concerning DC programming and DCA. The fundamentals of DC analysis, DC optimization such as DC functions, DC duality, global and local optimality in DC optimization are presented. After that, the presentation focuses on DCA is introduced, with the principle, the computation and the interpretations of DCA. In section 1.3, the presentation was devoted to how to use DCA and DCA combined with Branch and Bound technique to solve Mixed 0-1 Integer Linear Programming. The techniques which aim to find a good initial point and to restart DCA are also presented. This is an efficient method for solving MILP 0-1 and will be used frequently in the following chapters. Section 1.4 presents a short introduction of Proximal decomposition method and how to use this approach to solve convex optimization problems. When it comes to convex optimization problems with complex feasible sets, the proximal decomposition algorithm and the proximal copy decomposition algorithm were introduced to handle these problems. It will be useful in Chapter 2 to combine with DCA for solving QoS routing problems in large dimensions.

Part II

ROUTING PROBLEMS

Chapter 2

Solving QoS Routing problems using DCA

This chapter devotes to solve QoS routing problem via DC programming and DCA. Firstly, we analyzed the QoS routing problem from the classical Unicast QoS routing problems to the complex Multicast and Many to Many multicast QoS routing problem. We introduced five problems in Unicast, two problems in Multicast, two problems in Many to Many multicast QoS routing and their mathematical formulation. Secondly, based on DC programming, DCA and Proximal decomposition method, a solution method and its numerical simulation are investigated. The main content was concerned in [Ta et al., 2012c],[Ta et al., 2010b] and [Ta et al., 2010a].

The Unicast (resp. Multicast) Quality of Service (QoS) routing emphasizes to find paths (resp. a set of paths) from a source node to a destination node (resp. a set of destination nodes) satisfying the QoS requirements. In this paper, we consider five problems in Unicast QoS Routing called the MCP (Multi-Constrained Path) problems, the MCOP (Multi-Constrained Optimal Path) problems and two problems in Multicast QoS Routing called the MCM (Multi-Constrained Mutlicast Tree) problems and the MCOM (Multi-Constrained Optimal Multicast Tree) problems. After that, we consider an extension of Multicast QoS routing problem called Many to Many Multicast QoS routing problems, which problem deals with routing from multi-source nodes to multi-destination node sets satisfying the QoS constraints. It is a nonconvex program with a huge number of constraints problem. They are all NP-complete. We first formulate these problems as Concave Quadratic Program (CQP) or Binary Integer Linear Program (BILP) then investigate a new solution methods based on DC (Difference of Convex functions) programming and DCA (DC Algorithms). The numerical results for solving Unicast and Multicast QoS routing problem are compared with CPLEX, the best solver for BILP. To specify in Many to many Multicast QoS routing problems, because of large-scale problem, to handle the large number of constraints, we introduce the proximal decomposition technique in DCA to tackle convex subprograms at each iteration. Preliminary numerical simulations are reported to show the efficiency of our customized DCA and the quality of computed solutions.

2.1 Introduction

In recent years, transmissions of multimedia content over the communication networks have many challenges. Therefore, these transmissions might work properly, QoS measures like bandwidth, delay, jitter and packet loss, etc., need to be controlled. These issues give rise to the problem of routing in the network where the requirements of QoS are satisfied.

In general, the routing includes two entities: the routing protocol and the routing algorithm. The routing protocol has the task of capturing the state of the network and its available network resources, and of disseminating this information throughout the network. The routing algorithm uses this information to compute an optimal path from a source node to a destination node.

The Unicast QoS Routing Problem consists of finding an optimal path from a source node to a destination node subject to QoS constraints (e.g., time delay, cost or packets loss constraints). It is well-known that this problem is NP-hard [Yuan and Liu, 2001]. The Multicast QoS Routing aims to find a set of paths from a source node to a set of destination nodes satisfying the QoS requirements. It is also proved that this problem is NP-complete [Kuipers and Mieghem, 2002].

The Routing problems become more complex as far as we consider mobile networks or hybrid networks, because of dynamic topology and real time routing procedure.

As an example, we consider a scenario in Multicast routing problem, such as we are staying in a car parking place. The mobile services are provided in each moving car, equipped with a mobile device, via a car service center likes in the car parking place. There are m cars sending their requests to a mobile car service center, they need help to find the route to go to their destinations under the travel time constraint, the less latency traffic jam, the jitter time delay constraint, the travel cost (same sources, different destinations, considering local constraints to each mobile vehicle). So, based on the temporary update data of the network state, the mobile car service system has to calculate the route and given the answer for each car in a few seconds. In this context, we need a centralized and efficient algorithm to calculate the routes.

Many approaches for solving the Unicast QoS routing problems have been proposed (e.g., [Kuipers et al., 2002], [Liu and Ramakrushnam, 2001],[Mieghem and Kuipers, 2003], [Mieghem and Kuipers, 2004], etc.). Most of the existing approaches are based on classical methods for solving the shortest path problem such as Dijkstra or Ford-Bellman algorithms (e.g.,[Chen and Nahrstedt, 1998],[Mieghem and Kuipers, 2003],[Yuan and Liu, 2001]). Other approaches are based on network flow algorithms (e.g., [Orda and Sprintson, 2004]). In addition, there are several polynomial ε-approximate solution methods for solving these problems (e.g.,[Hassin, 1992],[Lorenz et al., 2000],[Warburton, 1987]).

For solving the Mutlicast QoS routing problems, many approaches have been proposed (e.g.,[Ali et al., 2008], [Kuipers and Mieghem, 2002],[Zhang et al., 2009], [Wang and Hou, 2000], etc.). The existing approaches are usually based on the classical methods for solving the shortest path tree problem (based on the Dijkstra and Ford-Bellman algorithms) (e.g., [Cormen et al., 1997],[Yuan and Liu, 2001]). Several heuristic methods are introduced [Blokh and Gutin, 1995],[Chen and Nahrstedt, 1998], [Korkmaz et al., 2002], [Korkmaz and Krunz, 2001], [Lorenz and Danny, 2001]... Other approaches are improved from the minimum spanning tree algorithms, such as extension of Prim algorithm (e.g., [Cormen et al., 1997], [Gallager et al., 1983]), or developed from Steiner tree problems (e.g., [Bauer, 1996],[Hwang, 1992], [Salama, 1996]).

For solving Many to Many multicast QoS routing problem, almost of the existing method are extension of method for solving Multicast QoS routing problem or heuristic method (see [Paul and Raghavan, 2002], [Tyan et al., 2003]). However, as father as we know, because of the large-scale problems, the solution methods for solving these problems are seldom investigated from a mathematical formulation, they usually use the properties of an optimal Multicast tree or an optimal Many to many multicast tree to construct an heuristic or approximate solution.

In this chapter, a new approach for solving the Unicast, Multicast and Many to Many Multicast QoS routing problems is provided. We have formulated five Unicast QoS routing problems, two Multicast and two Many to Many Multicast QoS routing problems in the form of CQP and BILP problems for which DC programming and DCA are investigated. Moreover, we consider large-scale problems for which we use DCA and Proximal decomposition technique to show the efficient of our approach.

As a result in Chapter 1.3, we know how to adapt this approach for solving the unicast and multicast QoS routing problems formulated as BILP. The proposed DCA for BILP

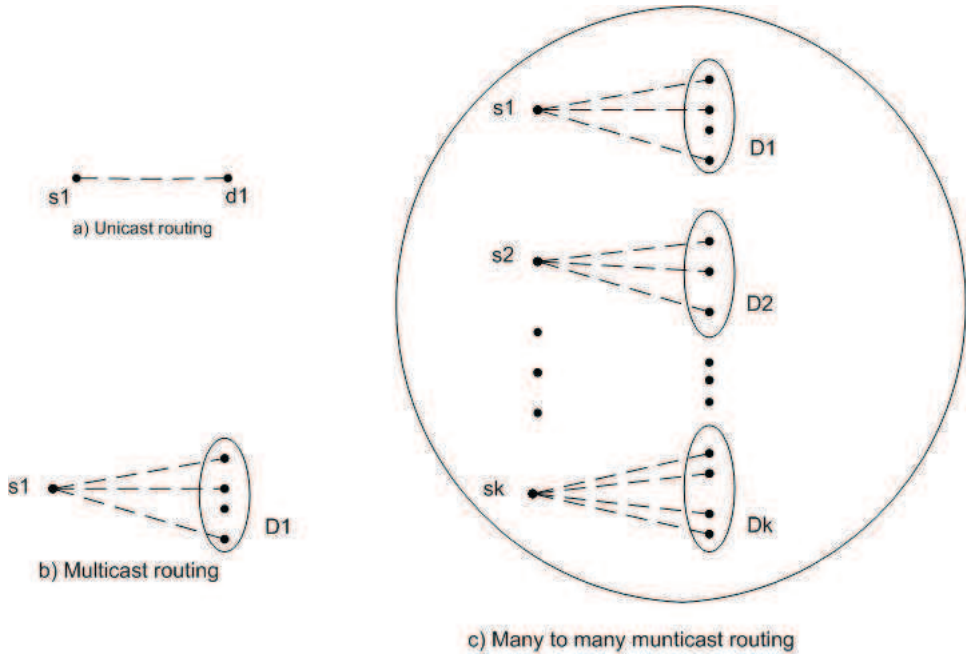

Figure 2.1: An illustration of Unicast, Multicast and Many to many multicast routings.

enjoys several advantages: it converges to a local (integer) solution after a finitely many iterations, and requires only the solution of a few number of linear programs. Moreover, it is worth to mention that, although the DCA is a continuous approach working on a continuous domain, it provides an integer solution. This is unusual in continuous approaches and is original property of the proposed DCA. The efficiency of DCA is compared with CPLEX, the best solver for BILP. The computational results on several test problems show that DCA is an efficient algorithm for solving BILP.

The efficiency of DCA combined with Proximal decomposition method is show that we can find a feasible solution of large-scale problems while CPLEX 12.2 (the best solver for BILP and Linear problem) cannot solve BILP problem, even CPLEX 12.2 cannot solve the Linear relaxation of BILP problem.

2.2 Problem statement and mathematical formulation

In this paper, the network-state information is assumed to be temporarily static, and this information is known at each node. The second task in unicast (resp. multicast) QoS routing is considered, namely computing a path (resp. a set of paths) with given multiple QoS constraints.

A network topology can be expressed as a weighted graph $G = (V, E)$, where V is node set, E is arc set. $|V|$ is the number of nodes and $|E|$ is the number of arcs in the network. Generally, supposed that there is only one arc between a pair of nodes in the network. The related parameters could be set to describe the current situation of arc. The number of QoS measures are denoted by p. Each arc is characterized by a p-dimensional arc weight vector. Each vector composes p non-negative QoS weight components which are denoted by $w_i(u, v), i = 1, \cdots, p, (u, v) \in E$.

The QoS measures of a path can either be additive, multiplicative, or min/max. The path weight of additive measures (e.g., time delay or cost) equals to the sum of the QoS weights of the arcs on the path. The multiplicative measures (e.g., probability packet loss) can be transformed into additive measures by using the logarithm function. The path

weight of min/max measures (e.g., min/max bandwidth) presents the minimum/maximum of the QoS weights defining the path. The constraints on min/max QoS measures can easily be settled by omitting all arcs (or disconnected nodes), which do not satisfy the requested QoS constraints. In practice, the constraints on additive QoS measures are more difficult. In this paper, QoS measures are assumed to be additive.

Definition 2.1 *Considering a path P of G composed of a set of links, the i^{th} weight w_i of P is defined as:*

$$w_i(P) = \sum_{(u,v)\in P} w_i((u,v)). \tag{2.1}$$

The vector weight of the path is also defined as:

$$w(P) = (w_1(P), w_2(P), \cdots, w_p(P)).$$

Definition 2.2 *(Multi-Constrained Path (MCP) Problem [Kuipers and Mieghem, 2005]) Consider a network $G = (V, E)$. Each arc $(u, v) \in E$ is specified by p additive QoS weights $w_i(u, v) \geq 0, i = 1, \cdots, p$. Given p-vector constraints $L = (L_1, L_2, \cdots, L_p)$, the problem consists of finding a path P from a source node s to a destination node t such that:*

$$w(P) \leq L.$$

A path P that satisfies the QoS constraints is called a feasible path. The Multi-Constrained Optimal Path (MCOP) problem aims to find the optimal path. If the QoS constraints are eliminated from the MCP and the MCOP, the resulting problems can be solved by Dijkstra or Ford-Bellman algorithms. However, the MCP and the MCOP are NP-hard problems.

2.2.1 Unicast QoS Routing problem

In this subsection, we introduce the formulation of five problems in Unicast QoS routing. Problem 1 consists of finding a feasible path. Problem 2 is to find an optimal path. Problem 3 and Problem 4 aim to find, respectively, k-arc disjoint paths and the maximal number of arc disjoint paths such that the QoS constraints are satisfied. Problem 5 consists of finding k feasible arc disjoint paths $P_1, \cdots, P_k$, each of them satisfying the QoS constraints.

Problem 1: *Finding a feasible path P.*

Define the binary variable y_{ij} as:

$$y_{ij} = \begin{cases} 1 \text{ if } (i,j) \in P, \\ 0 \text{ otherwise.} \end{cases}$$

Then a path from s to t must satisfy the following constraints:

$$\sum_{(s,v)\in E} y_{sv} - \sum_{(u,s)\in E} y_{us} = 1,$$

$$\sum_{(u,v)\in E} y_{uv} - \sum_{(v,\ell)\in E} y_{v\ell} = 0, \ \forall v \in N\backslash\{s,t\},$$

$$\sum_{(t,v)\in E} y_{tv} - \sum_{(u,t)\in E} y_{ut} = -1.$$

The QoS constraints can be formulated as:

$$\sum_{(u,v)\in E} w_i(u,v) y_{uv} \leq L_i, \ i = 1, \cdots, p. \tag{2.2}$$

If the value of $w_i(u,v)$ is fixed to each arc $(u,v) \in E$ (e.g., the cost for sending one message or one unit data on the arc (u,v), or the time delay on the arc (u,v)), then the QoS constraints are linear constraints. When $w_i(u,v)$ is also a variable, this problem becomes more complex.

Let us define a_{ij} as follows:

$$a_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E, \\ -1 \text{ if } (j,i) \in E, \\ 0 \text{ otherwise.} \end{cases}$$

Furthermore, let $A = (a_{ij})$ (resp. $W = (w_i(u,v))$) be a $m \times n$ (resp. $p \times n$) matrix, let $b = (1, 0, \cdots, 0, -1)^T \in \mathbb{R}^m$, $y = (y_{ij})^T_{(i,j)\in E} \in \mathbb{R}^n$, and $L = (L_1, \cdots, L_p)^T \in \mathbb{R}^p$.

Then, the QoS constraints (2.2) become $Wy \leq L$. Problem 1 consists of finding $y \in \{0,1\}^n$ such that $Ay = b$ and $Wy \leq L$.

We define a concave function $p(y)$ as follows:

$$p(y) = \sum_{(i,j)\in E} y_{ij}(1 - y_{ij})^1.$$

We see that $p(y) \geq 0 \; \forall y \in [0,1]^n$ and $y \in \{0,1\}^n$ iff $p(y) = 0$.

Finally, Problem 1 can be formulated as a CQP in the following:

$$(\mathbf{P1}) \begin{cases} 0 = \min p(y) \\ Ay = b, \; Wy \leq L, \\ y \in [0,1]^n. \end{cases}$$

Problem 2: *Finding an optimal path P satisfying the QoS constraints.*

Problem 2 aims to find a feasible path with a minimum number of arcs under the QoS constraints. Clearly, by adding an objective function into (P1) we obtain the mathematical formulation of Problem 2 in the form of a BILP:

$$(\mathbf{P2}) \quad \min \left\{ \sum_{(i,j)\in E} y_{ij} : Ay = b \quad \text{and } Wy \leq L, y \in \{0,1\}^n \right\}.$$

Problem 3: *Finding k feasible arc disjoint paths which satisfy the QoS constraints.*

Let us define the binary variable y_{ij} as:

$$y_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \text{ belongs to the set of } k \text{ arc disjoint paths,} \\ 0 \text{ otherwise.} \end{cases}$$

Let $b_1 = (k, 0, \cdots, 0, -k)^T \in \mathbb{R}^n$. Problem 3 aims to find $y \in \{0,1\}^n$ such that $Ay = b_1$ and $Wy \leq L$, with the matrixes A, W, L being defined above.

Problem 3 can be formulated in a similar form with Problem 1,

$$(\mathbf{P3}) \begin{cases} 0 = \min p(y) \\ Ay = b_1, \; Wy \leq L, \\ y \in [0,1]^n. \end{cases}$$

Problem 4: *Finding the maximal number of arc disjoint paths from s to t which satisfy the QoS constraints.*

[1] Another definition of function $p(y)$ as $p(y) = \sum_{(i,j)\in E} \min\{y_{ij}, 1 - y_{ij}\}$ is also considered with similar properties.

Let $d = (d_{(i,j)\in E})^T \in \mathbb{R}^n$, where

$$d_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \text{ is outgoing arc from } s, \\ -1 \text{ if } (i,j) \in E \text{ is ingoing arc to } s, \\ 0 \text{ otherwise.} \end{cases}$$

Problem 4 (P4) can be then formulated as

$$\max\langle d, y\rangle$$

subject to

$$\sum_{(s,v)\in E} y_{sv} - \sum_{(u,s)\in E} y_{us} = k,$$

$$\sum_{(t,v)\in E} y_{tv} - \sum_{(u,t)\in E} y_{ut} = -k,$$

$$\sum_{(u,v)\in E} y_{uv} - \sum_{(v,\ell)\in E} y_{v\ell} = 0,\ \forall v \in N\backslash\{s,t\},$$

$$Wy \leq L \text{ and } y \in \{0,1\}^n.$$

In wireless networks, the reliability (defined as the success packet delivery) is a key factor. In the case where an arc changes and fails, the network is unreliable. The arising problem is to find an alternative path when the initial one is damaged.

Problem 5: *Finding k feasible arc disjoint paths $P_1, \cdots, P_k$, each path satisfies the QoS constraints.*

Define the binary variable $y_{ij\ell}$ as:

$$y_{ij\ell} = \begin{cases} 1 \text{ if } (i,j) \in P_\ell,\ \ell = 1, \cdots, k, \\ 0 \text{ otherwise.} \end{cases}$$

Then, similarly to Problem 1, we have the following constraints:

$$\sum_{(s,v)\in E} y_{sv\ell} - \sum_{(u,s)\in E} y_{us\ell} = 1, \tag{2.3}$$

$$\sum_{(u,v)\in E} y_{uv\ell} - \sum_{(v,q)\in E} y_{vq\ell} = 0,\ \forall v \in N\backslash\{s,t\}, \tag{2.4}$$

$$\sum_{(t,v)\in E} y_{tv\ell} - \sum_{(u,t)\in E} y_{ut\ell} = -1. \tag{2.5}$$

On the other hand, the next constraint ensures that each arc $(i,j) \in E$ belongs to at most one path P_ℓ ($\ell = 1, \cdots, k$):

$$\sum_{\ell=1}^{k} y_{ij\ell} \leq 1, \text{for each } (i,j) \in E. \tag{2.6}$$

The QoS constraints for each path P_ℓ ($\ell = 1, \cdots, k$) are expressed as

$$\sum_{(u,v)\in P_\ell} w_i(u,v) \leq L_i,\ i = 1, \cdots, p. \tag{2.7}$$

Problem 5 deals with finding $y_{ij\ell} \in \{0,1\}((i,j) \in E,\ \ell = 1, \cdots, k)$ such that (2.3) – (2.7) are satisfied.

Problem 5 can be formulated similarly to Problem 1 in the form of CQP.

We observe that (P1), (P3) and (P5) are MCP problems while (P2) and (P4) are of the form MCOP.

Remark: The MCP problems and MCOP are formulated, respectively, as QCPs and BILPs.

2.2.2 Multicast QoS Routing problem

To specify multicast QoS routing, the same assumption presented above are adopted. The constraint vector L represents the limits allowed for the path weights from the source node to every member of the destination node set. It represents the limits for end-to-end values and not for the sum of values in the multicast structure.

Multicast QoS routing problem consists in finding a set of paths from a source node s to ℓ destination nodes t_j $(j = 1, \cdots, \ell)$. In traditional multicast routing, this set of paths corresponds to a tree but in multicast QoS routing it is not compulsorily a tree. In general case, it corresponds to a sub-graph $T = (M, H)$ of G. T is regarded as a set of paths from s to t_j $(j = 1, \cdots, \ell)$ which use the arcs in H.

Under such assumption, the Multi-Constrained Multicast (MCM)problems were introduced in [Kuipers and Mieghem, 2002].

Definition 2.3 *((MCM) [Kuipers and Mieghem, 2002]) Let $p_T(s, t_j)$ be the path from source node s to destination node t_j. $T(s, D)$ is a multicast tree. $s \in V$ is the source node in multicast tree. $D = \{t_1, t_2, \cdots, t_\ell\} \subset V \backslash \{s\}$ is the destination node set in multicast tree. The problem consists in finding a multicast tree $T(s, D)$ such that*

$$w(p_T(s, t_j)) \leq L_j, \ \forall j = 1, 2, \cdots, \ell,$$

where $w(p_T(s, t_j))$ is calculated as (2.1) and L_j is a p-vector weight constraints corresponding destination t_j.

A multicast tree T that satisfies the QoS constraints is called *a feasible multicast tree.* The Multi-Constrained Optimal Multicast Tree (MCOM) problem consists of finding an optimal multicast tree. The objective function of the MCOM problem can be defined as in [Kuipers and Mieghem, 2002], or as total arc-cost in the multicast tree ([Zhang et al., 2009]),$\cdots$. It is proved in [Kuipers and Mieghem, 2002] that MCM is a NP-complete problem.

We introduce now the formulation of two problems in Multicast QoS routing as CQP and BILP problems. Problem 6 consists in finding a feasible multicast tree. Problem 7 aims to find an optimal multicast tree in which the objective function is equal to the sum of the arc-costs in the multicast tree.

Problem 6: *Finding a feasible multicast tree T(s, D).*

The capacity of each arc $(u, v) \in E$ is denoted by $e(u, v) \in \mathrm{Z}^+$, where Z^+ is the set of positive integer numbers.

Define the binary variable y_{ijk} as:

$$y_{ijk} = \begin{cases} 1 \text{ if arc } (i, j) \in p_T(s, t_k), \ k = 1, \cdots, \ell, \\ 0 \text{ otherwise.} \end{cases}$$

Then, we have the following constraints:

$$\sum_{(s,v)\in E} y_{svk} - \sum_{(u,s)\in E} y_{usk} = 1, \ k = 1, \cdots, \ell, \tag{2.8}$$

$$\sum_{(u,v)\in E} y_{uvk} - \sum_{(v,q)\in E} y_{vqk} = 0, \ \forall v \in N \backslash \{s, D\}, \tag{2.9}$$

$$\sum_{(t,v)\in E} y_{tvk} - \sum_{(u,t)\in E} y_{utk} = -1, \forall t \equiv t_k,\ k = 1, \cdots, \ell. \tag{2.10}$$

In the other hand, the constraints ensure that the total transfer packets in each arc $(i,j) \in E$ is less or equal the capacity of the arc (i,j) can be described as:

$$\sum_{k=1}^{\ell} y_{ijk} \leq e(i,j),\ \text{for each } (i,j) \in E. \tag{2.11}$$

The QoS constraints for each path $p_T(s,t_j)$

$$w(p_T(s,t_j)) \leq L_j,\ \forall j = 1,2,\cdots,\ell,$$

can be expressed as:

$$\sum_{(i,j)\in E} w_q(i,j) y_{ijk} \leq L_{qk},\ \forall q = \overline{1,p};\ k = \overline{1,\ell}. \tag{2.12}$$

Let us define a matrix $A = (a_{ij})$, a matrix $W = (w_i(u,v))$ and a vector $b = (1,0,\cdots,0,-1)^T \in \mathbb{R}^m$, as in the Problem 1. Furthermore, let $y^k = (y_{ijk})^T_{(i,j)\in E} \in \mathbb{R}^n$ $(k = 1,\cdots,\ell)$ and $L_j = (L_{1j},\cdots,L_{pj})^T \in \mathbb{R}^p$ $(j = 1,\cdots,\ell)$.

The constraints (2.8)-(2.10) and QoS constraints (2.12) can be rewritten respectively as $Ay^k = b$ and $Wy^k \leq L_k\ \forall k = 1,\cdots,\ell$.

Let us set $y = (y^1, y^2, \cdots, y^\ell)^T \in \mathbb{R}^{n\ell}$, a vector capacity $e = (e(i,j))_{(i,j)\in E}$ and a matrix $M_{n\times n\ell}$ such that the constraints (2.11) can be expressed as $My \leq e$.

Problem 6 is to find $y \in \{0,1\}^{n\ell}$ satisfied $Ay^k = b,\ Wy^k \leq L_k,\ \forall k = 1,\cdots,\ell$ and $My \leq e$.

A concave function $p(y)$ is defined as follows:

$$p(y) = \sum_{(i,j)\in E, k=1,\cdots,\ell} y_{ijk}(1 - y_{ijk}).$$

We can see that $p(y) \geq 0\ \forall y \in [0,1]^{n\ell}$ and $y \in \{0,1\}^{n\ell}$ iff $p(y) = 0$.

Hence, Problem 6 can be formulated as a CQP:

$$(\mathbf{P6}) \begin{cases} 0 = \min p(y) \\ Ay^k = b,\ \forall k = 1,\cdots,\ell, \\ Wy^k \leq L_k,\ \forall k = 1,\cdots,\ell, \\ My \leq e, \\ y \in [0,1]^{n\ell}. \end{cases}$$

Problem 7: *Finding an optimal multicast tree T(s, D).*

The cost value of each arc $(i,j) \in E$ is given by c_{ij}. The total cost function is defined by

$$f(T) = \sum_{(i,j)\in T} c_{ij}.$$

Problem 7 deals with finding a multicast tree which minimizes total cost function and satisfies QoS constraints.

The binary variable y_{ijk} is defined as in Problem 6. All the constraints and definitions in Problem 6 are considered in Problem 7.

On the other hand, let us defined the binary variable x_{ij} by

$$x_{ij} = \begin{cases} 1 \text{ if arc } (i,j) \in T(s,D), \\ 0 \text{ otherwise.} \end{cases}$$

The constraints which present the value of x_{ij} depending on the arc $(i,j) \in T$ can be expressed:

$$x_{ij} \leq \sum_{k=1}^{\ell} y_{ijk} \leq e(i,j)x_{ij}. \tag{2.13}$$

The total cost function (the objective function) can be calculated as

$$f(T) = \sum_{(i,j)\in E} c_{ij}x_{ij}.$$

Let us set $x = (x_{ij})^T_{(i,j)\in E}, c = (c_{ij})^T_{(i,j)\in E} \in \mathbb{R}^n$, a matrix $D = \text{diag}(e(i,j)_{(i,j)\in E})$. So, the constraints (2.13) can be rewritten as $x \leq My \leq Dx$. It is easy to see that the constraint (2.13) covers the constraint (2.11).

Problem 7 can be formulated as a BILP:

$$(\mathbf{P7}) \begin{cases} \min\langle c, x\rangle \\ Ay^k = b,\ \forall k = 1, \cdots, \ell, \\ Wy^k \leq L_k,\ \forall k = 1, \cdots, \ell, \\ x \leq My \leq Dx, \\ x \in \{0,1\}^n,\ y \in \{0,1\}^{n\ell}. \end{cases}$$

Problem 6 and Problem 7 are formulated, respectively, as CQP and BILP.

2.2.3 Many to Many Multicast QoS Routing problem

Definition 2.4 *Let $S = \{s_1, \cdots, s_K\}$ (resp. $D = \{D_1, \cdots, D_K\}$) be the set of source nodes (resp. the set of destination node sets, source node s_i corresponding with destination node set D_i). Many to many multicast QoS routing deals with finding K multicast trees $T_i(s_i, D_i)$, $(i = 1, \cdots K)$, each of them satisfies the QoS constraints and the capacity constraints in global network (e.g. limited of capacity).*

We introduce now the formulation of Many to many multicast QoS routing as CQP or BILP problems. Problem 8 consists in finding a feasible many to many multicast trees. Problem 9 deals with the finding of an optimal many to many multicast trees in which the objective function is equal to sum of arc-cost in the many to many multicast trees.

Problem 8: *Finding a feasible many to many multicast tree $T_i(s_i, D_i)$, $i = 1, \cdots, K$.*

The capacity of each arc $(u,v) \in E$ is denoted by $e(u,v) \in \mathrm{Z}^+$.

Define the binary variable y_{tijk} as:

$$y_{tijk} = \begin{cases} 1 \text{ if arc } (i,j) \in p_{T_t}(s_t, t_k), t_k \in D_t, \\ 0 \text{ otherwise.} \end{cases}$$

Let us set $|D_i| = \ell_i,\ i = 1, \cdots, K$ (the number of nodes in D_i).

Then, for $t = 1, \cdots, K,\ s \equiv s_t$, and $k = 1, \cdots, \ell_t,$, we have the following constraints (which aims to find a path from s_t to $t_k \in D_t$):

$$\sum_{(s,v)\in E} y_{tsvk} - \sum_{(u,s)\in E} y_{tusk} = 1, \tag{2.14}$$

$$\sum_{(u,v)\in E} y_{tuvk} - \sum_{(v,q)\in E} y_{tvqk} = 0,\ \forall v \in N\backslash\{s, t_k\}, \tag{2.15}$$

$$\sum_{(o,v)\in E} y_{tovk} - \sum_{(u,o)\in E} y_{tuok} = -1, \forall o \equiv t_k \in D_t. \tag{2.16}$$

In the other hand, the constraints ensure that the total transfer packets in each arc $(i,j) \in E$ is less or equal the capacity of the arc (i,j) can be described as:

$$\sum_{t=1}^{K} \sum_{k=1}^{\ell_t} y_{tijk} \leq e(i,j), \text{ for each } (i,j) \in E. \tag{2.17}$$

The QoS constraints for each path $p_{T_i}(s_i,t_j)(t_j \in D_i\ i = 1,\cdots,K)$

$$w(p_{T_i}(s_i,t_j)) \leq L_{ij},\ \forall t_j \in D_i,$$

can be expressed as:

$$\sum_{(i,j)\in E} w_q(i,j) y_{tijk} \leq L_{qtk},\ \forall q = \overline{1,p};\ t_k \in D_t. \tag{2.18}$$

Let us define a_{ij} as follows:

$$a_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E, \\ -1 \text{ if } (j,i) \in E, \\ 0 \text{ otherwise.} \end{cases}$$

Furthermore, let $A = (a_{ij})$ (resp. $W = (w_i(u,v))$) be a $m \times n$ (resp. $p \times n$) matrix, let $b = (1,0,\cdots,0,-1)^T \in \mathbb{R}^m$, $y^{tk} = (y_{tijk})^T_{(i,j)\in E} \in \mathbb{R}^n$ $(t = 1,\cdots K,\ k = 1,\cdots,\ell_t)$ and $L_{tj} = (L_{1tj},\cdots,L_{ptj})^T \in \mathbb{R}^p$ $(j = 1,\cdots,\ell_t)$.

The constraints (2.14)-(2.16) and QoS constraints (2.18) can be rewritten respectively as $Ay^{tk} = b$ and $Wy^{tk} \leq L_{tk}\ \forall t = 1,\cdots,K,\ k = 1,\cdots,\ell_t$.

Let us set $y^t = (y^{t1}, y^{t2},\cdots,y^{t\ell})^T \in \mathbb{R}^{n\ell_t}$, vector capacity $e = (e(i,j))_{(i,j)\in E}$ and matrix $M_{t(n\times n\ell_t)}$ such that the constraints (2.17) can be expressed as $M_t y^t \leq e$.

Problem 8 consists of finding $y^t \in \{0,1\}^{n\ell_t}$ satisfied $Ay^{tk} = b$, $Wy^{tk} \leq L_{tk}$, $\forall t = 1,\cdots,K$, $k = 1,\cdots,\ell_t$ and $M_t y^t \leq e$. Let us set $y = (y^1,\cdots,y^K)$. We see that $y^t \in \{0,1\}^{n\ell_t}, t = 1,\cdots,K$ is equivalent to $p(y) = 0$ where $p(y) = \sum y_i(1-y_i)$ and $p(y) \geq 0\ \forall y^t \in [0,1]^{n\ell_t}$, $t = 1,\cdots,K$. Hence, we define matrix A_t, W_t $(t = 1,\cdots,K)$ such that Problem 8 can be formulated as:

$$(\mathbf{P8}) \begin{cases} 0 = \min p(y) \\ A_t y^t = b_t,\ W_t y^t \leq L_t, \\ y^t \in [0,1]^{n\ell_t},\ M_t y^t \leq e, \\ \forall t = 1,\cdots,K. \end{cases}$$

Problem 9: *Finding an optimal many to many multicast tree $T_i(s_i,D_i)$ $(i = 1,\cdots,K)$.*

The cost value of each arc $(i,j) \in E$ is given by c_{ij}. The total cost function is defined by $f(T) = \sum_{t=1}^{K} \sum_{(i,j)\in T_t} c_{ij}$. Problem 9 deals with finding a many to many multicast tree which minimizes total cost function and satisfies QoS constraints.

We define the binary variable y_{tijk} as the same as in Problem 8. All the constraints and definitions in Problem 8 are taken into account in Problem 9.

In the other hand, let the binary variable x_{tij} be:

$$x_{tij} = \begin{cases} 1 \text{ if arc } (i,j) \in T(s_t,D_t), t = 1,\cdots,K \\ 0 \text{ otherwise.} \end{cases}$$

Hence,for $t = 1,\cdots,K$, the constraints which present the value of x_{tij} depending on the arc $(i,j) \in T_t$ can be expressed:

$$x_{tij} \leq \sum_{k=1}^{\ell_t} y_{tijk} \leq e(i,j) x_{tij}. \tag{2.19}$$

The global capacity constraints in the network can be expressed as:

$$\sum_{t=1}^{K} x_{tij} \leq e(i,j),\ \forall (i,j) \in E. \tag{2.20}$$

The total cost function (the objective function) can be calculated

$$f(T) = \sum_{t=1}^{K} \sum_{(i,j)\in E} c_{ij} x_{tij}.$$

Let us set $x^t = (x_{tij})^T_{(i,j)\in E}, c = (c_{ij})^T_{(i,j)\in E} \in \mathbb{R}^n$, matrix $D = \text{diag}(e(i,j)_{(i,j)\in E})$. So, the constraint (2.19) can be rewritten as $x^t \leq M_t y^t \leq D x^t,\ t = 1, \cdots, K$. It is easy to see that the constraint (2.19) covers the constraint (2.17). So, Problem 9 can be formulated as:

$$(\mathbf{P9}) \begin{cases} \min \sum_{t=1}^{K} \langle c, x^t \rangle \\ A_t y^t = b_t,\ W_t y^t \leq L_t, \\ x^t \leq M_t y^t \leq D x^t, \\ \sum_{t=1}^{K} x_{tij} \leq e(i,j), \forall (i,j) \in E \\ x^t \in \{0,1\}^n,\ y^t \in \{0,1\}^{n\ell_t}, \\ \forall t = 1, \cdots, K. \end{cases}$$

Problem 8 and Problem 9 are formulated as CQP and BILP, respectively.

In the sequence, we investigate a new approach based on DCA for solving these problems in the form of BILP. The proposed algorithm for solving BILP problems are obtained as the results of Chapter 1.3.

2.3 Solving Many to Many Multicast Tree problem by DCA

In this section, we consider the problem 9 (P9). Because the problem 9 is the most complex problems with huge number of variables and constraints. The other problems in the form of BILPs can solve with the same algorithm which is presented following.

By using an exact penalty result, we can reformulate problem (P9) in the form of a concave minimization program. The exact penalty technique aims at transforming the original problem (P9) into a more tractable equivalent DC program.

Let us set $z = (x, y)$, matrix $\widetilde{A}$ and vector $\widetilde{b}$ are defined such that the constraints (2.14)-(2.16), (2.18), (2.19) and (2.20) can be expressed as $\widetilde{A}z \leq \widetilde{b}$.

Let

$$\mathcal{K} := \{z \in \mathbb{R}^N : \widetilde{A}z \leq \widetilde{b},\ z \in [0,1]^N\}.$$

The feasible set of (P2) is then

$$S = \{z : z \in \mathcal{K},\ z \in \{0,1\}^N\}.$$

Let us consider the function $p : \mathbb{R}^N \to \mathbb{R}$ defined by:

$$p(z) = \sum_{i=1}^{N} z_i(1 - z_i).$$

It is clear that $p(z)$ is concave and finite on $\mathcal{K}$, $p(z) \geq 0\ \ \forall z \in K$ and that:

$$\big\{z : z \in S\big\} = \big\{z : z \in \mathcal{K}, p(z) \leq 0\big\}. \tag{2.21}$$

Hence problem (P9) can be rewritten as:

$$\min\Big\{\sum_{i=1}^{N} c_i z_i : z \in \mathcal{K}, p(z) \leq 0\Big\}. \tag{2.22}$$

From the penalty results in Theorem 1.6 we get, for a sufficiently large number η ($\eta > \eta_0$), the equivalent concave minimization problem (P9):

$$\min\{f_\eta(z) := \sum_{i=1}^{N} c_i z_i + \eta p(z) : z \in \mathcal{K}\},$$

which is a DC program of the form:

$$\min\big\{g(z) - h(z) : z \in \mathbb{R}^N\big\}, \tag{2.23}$$

where:

$$g(z) = \chi_{\mathcal{K}}(z) \text{ and } h(z) = -f_\eta(z) = -\sum_{i=1}^{N} c_i z_i - \eta p(z).$$

We have successfully transformed an optimization problem with integer variables into its equivalent form with continuous variables. Notice that (2.23) is a polyhedral DC program where g is a polyhedral convex function (i.e., the pointwise supremum of a finite collection of affine functions).

DCA applied to the DC program (2.23) consists of computing, at each iteration k, the two sequences $\{z^k\}$and $\{v^k\}$ such that $v^k \in \partial h(z^k)$ and z^{k+1} solves the next linear program of the form (P_k)

$$\begin{aligned} &\min\{g(z) - \langle z - z^k, v^k\rangle : z \in \mathbb{R}^N\} \\ &\Leftrightarrow \min\{-\langle z\ , v^k\rangle :\ z \in \mathcal{K}\}. \end{aligned} \tag{2.24}$$

From the definition of h, a subgradient $v^k \in \partial h(z^k)$ can be computed as follows:

$$v^k = \nabla h(z^k) = 2\eta z^k - \eta \mathbf{e} - c, \tag{2.25}$$

where $e = (1, \cdots, 1)^T \in \mathbb{R}^N$ and $c = (c_1, \cdots, c_N)^T$.

The DCA scheme applied to Problem (2.23) can be summarized as **Algorithm 4**.

Algorithm 4

Initialization:
Choose a initial point z^0, set $k = 0$;
Let $\varepsilon_1, \varepsilon_2$ be sufficiently small positive numbers;
Repeat
Compute v^k via (2.25);
Solve the linear program (2.24) to obtain z^{k+1};
$k \leftarrow k + 1$;
Until either $\|z^{k+1} - z^k\| \leq \varepsilon_1(\|z^k\| + 1)$ or $|f_\eta(z^{k+1}) - f_\eta(z^k)| \leq \varepsilon_2(|f_\eta(z^k)| + 1)$.

Theorem 2.1 *(Convergence properties of Algorithm DCA)*

- *DCA generates the sequence $\{z^k\}$ contained in $V(K)$ such that the sequence $\{f_\eta(z^k)\}$ is decreasing.*
- *The sequence $\{z^k\}$ converges to $z^* \in V(K)$ after a finite number of iterations.*

- *The point z^* is a critical point of Problem (2.23). Moreover if $z_i^* \neq \frac{1}{2}$ for all $i \in \{0, \ldots, n\}$, then z^* is a local solution to (2.23).*
- *For a number η sufficiently large, if at iteration r we have $z^r \in \{0,1\}^n$, then $z^k \in \{0,1\}^n$ for all $k \geq r$.*

Proof 2.1 *See Theorem 1.7.*

Remark: For solving CQP problems, we can apply DCA in a similar way, for instance, with Problem 1, we can choose a DC decomposition as $g(z) = \chi_K(z)$ and $h(z) = -p(z)$.

2.4 Totally unimodular matrices and Initial point for DCA

2.4.1 Totally unimodular matrices

A matrix A is called *totally unimodular matrix* if each subdeterminant of A is 0, +1 or -1. So, each element of A is 0, +1 or -1. The relation of integer linear program and totally unimodular matrix is presented as the following theorem:

Theorem 2.2 *Let A be a totally unimodular matrix and let b be an integral vector. Then the polyhedron $P := \{x : Ax \leq b\}$ is integral.*

Proof 2.2 *See Theorem 19.1 in [Schrijver, 1998].*

The fundamental characteristic of totally unimodular matrix is introduced as follows.

Theorem 2.3 *Let A be the matrix with entries 0, +1 or -1. The following are equivalent:*

(i) A is totally unimodular, i.e;, each square submatrix of A has determinant 0, +1 or -1;

(ii) for each integral vector b the polyhedron $\{x : x \geq 0, Ax \leq b\}$ has only integral vertices;

(iii) for all integral vector a, b, c, d the polyhedron $\{x : c \leq x \leq d, a \leq Ax \leq b\}$ has only integral vertices;

(iv) each collection of columns of A can be split into two parts so that the sum of the columns in one part minus the sum of the column in the other part is a vector with entries only 0, +1 or -1;

(v) each nonsingular submatrix of A has a row with an odd number of non-zero components;

(vi) the sum of the entries in any square submatrix with even row and column sums is divisible by four;

(vii) no square of A has determinant +2 or -2.

Proof 2.3 *See Theorem 19.3 in [Schrijver, 1998].*

Theorem 2.4 *For those QoS routing problem (Problem $1, 2, 3, 4, 5, 6, 8$), if we do not consider the QoS constraints then the matrices which present the constraints of those problems are totaly unimodular matrices.*

Proof 2.4 *For Problem 1, 2, 3, 4, the matrices are nodes-arcs adjacent matrices, then see Chapter 19 in [Schrijver, 1998].*

For Problem 5, we do partition the set of constraints such as: Set 1 consists of constraints (2.3),(2.4),(2.5), Set 2 includes constraints (2.6). For each collection of rows of the constraints, we can do partition to S1 and S2 (corresponding with the constraints belong to Set 1 or Set 2), by the property (iv) of Theorem 2.3 and the totaly unimodular property of matrix A and its transpose A^T is equivalent, then the constraints matrix of Problem 5 without QoS constraints is a totaly unimodular matrix.

Problem 6 and Problem 8 can do in a similar way than with Problem 5.

2.4.2 Initial point for DCA

As further as we know, when using DCA, the question of finding *an appreciate DC decomposition* and *a good initial point* for DCA are still open questions. Therefore, for each particular problem, we try to find a DC decomposition such that the calculation is *simple*. Moreover, because DCA is a local algorithm, then we also try to find a good initial point for DCA.

In QoS routing problem, we usually use the solution of the linear relaxation problem as an initial point of DCA. On the other hand, we also use the solution of the relaxed problem, in which the QoS constraints is removed. In this case, if the constraint matrix is a totaly unimodular matrix, then by solving a linear program we have a *near* optimal point to use as an initial point of DCA.

2.5 Proximal decomposition method for solving sub-convex problems

The (PDM) algorithm is used in our problem based on Proximal decomposition algorithm on the graph of a maximal monotone operator, which is introduced in [Mahey et al., 1995]. The main result is summarized on Chapter 1.4.

The problem consists in finding:

$$(x, y) \in \mathcal{A} \times \mathcal{A}^{\perp} \text{ such that } y \in \mathcal{T}(x), \tag{2.26}$$

where $\mathcal{A}$ is a subspace of a finite dimensional vector space X , $\mathcal{A}^{\perp}$ is the orthogonal subspace of $\mathcal{A}$ and $\mathcal{T}$ is a maximal monotone operator.

The algorithm consists in two separate step at each iteration:

- A proximal step is to regularize the objective function by adding a quadratic term depending depending on the previous primal-dual pair of solution.
- A projection step on the corresponding subspaces.

Note that, when f_0 is a l.s.c proper convex function on $\mathbb{R}^n$, the operator $\mathcal{T} \equiv \partial f_0$, then a vector x is a solution of (2.26) if and only if $0 \in \partial(\chi_{\mathcal{A}} + f_0)(x)$, where $\chi_{\mathcal{A}}(x)$ is the indicator function of x on $\mathcal{A}$. Following the theorem 23.8 in [Rockafellar, 1970], $0 \in \chi_{\mathcal{A}}(x) + f_0(x)$.

Because of $\partial\chi_{\mathcal{A}}(x) = \mathcal{A}^{\perp}$ then $x \in \mathcal{A}$ is a solution of (2.26) if and only if there exists $y \in \mathcal{A}^{\perp}$ subject to $y \in \partial f_0(x)$. That leads so to finding a couple $(x, y) \in (\mathcal{A}\times\mathcal{A}^{\perp})\cap Gr(\partial f_0)$.

When the dimension of Many to Many multicast QoS Routing problem is large, the number of variables and the number of constraints are very large. We propose to use Proximal decomposition for solving sub-convex problems (2.24) in each iteration of DCA. Our purpose is to divide these convex problems into many sub-problems with a smaller set of constraints and it is easier to solve. We consider the problem in the following (see problem (2.24)):

$$\min\{f_0(x) = \langle c^*, x\rangle, x \in \mathcal{K} = \cap_{i=1}^{K} C_i,\ C_i \text{ closed convex}\}, \tag{2.27}$$

where $c^* \equiv v^k$ in Problem (2.24) and C_i is defined by

$$\begin{aligned} C_1 = \{x = (u, v) : A_1 v^1 = b_1,\ W_1 v^1 \le L_1,\ u^1 \le M_1 v^1 \le D u^1, \sum_{t=1}^{K} u_{tij} \le e(i, j), \\ \forall (i, j) \in E,\ u^t \in \{0, 1\}^n,\ v^t \in \{0, 1\}^{n\ell_t},\ t = 1, \cdots, K.\}, \\ C_i = \{x = (u, v) : A_i v^i = b_i,\ W_i v^i \le L_i,\ u^i \le M_i v^i \le D u^i,\ u^t \in \{0, 1\}^n, \\ v^t \in \{0, 1\}^{n\ell_t},\ t = 1, \cdots, K.\},\ \forall i = 2, \cdots, K. \end{aligned}$$

Define $F(x_1, \cdots, x_K) = \sum_{i=1}^{K} f_i(x_i)$ where $f_i(x) \equiv f_0(x) \forall i = 1, \cdots, K$, and $C = C_1 \times \cdots \times C_K$. The below problem is taken into account:

$$\min\{F(X), X = (x_1, \cdots, x_K) \in C, x_1 = \cdots = x_K\}. \tag{2.28}$$

Let us set $\mathcal{A} = \{X = (x_1, \cdots, x_K) : \ x_1 = \cdots = x_K\}$ and $\mathcal{T} = \prod_{i=1}^{K} \partial(f_i + \chi_{C_i})$, then $\mathcal{A}^{\perp} = \{(y_1, \cdots, y_K) : \ y_1 + y_2 + \cdots + y_K = 0\}$. The problem (2.28) can be rewritten as:

$$\min\{(F + \chi_C)(X) : X \in \mathcal{A}\} \tag{2.29}$$

Following the results of *Proximal copy decomposition algorithm* and *Proximal scaled decomposition algorithm* in Chapter 1.4, the **PDΛ** algorithm for solving Problem (2.29) can be expressed as **Algorithm 5**.

Algorithm 5 PDΛ

Initialization:
$(X^0, Y^0) \in A \times A^{\perp}, \ \lambda > 0$
Repeat
If $(X_t, Y_t) \in Gr(T)$ **then STOP**
Else
$Z^t = X^t + \lambda Y^t$
Proximal step:
$U^t = (I + \mathcal{T})^{-1}(Z^t)$, it means that
$U_i^t = (I + \partial(f_i + \chi_{C_i}))^{-1}(Z_i^t)$, then
$U_i^t = \text{argmin } \{f_i(U_i) + \frac{1}{2\lambda}||Z_i^t - U_i||^2, U_i \in C_i\}$.
$V^t = \frac{1}{\lambda}(Z^t - U^t)$.
Projection step:
$(X^{t+1}, Y^{t+1}) = \text{Proj}_{A \times A^{\perp}}(U^t, V^t)$, it means that
$X^{t+1} = (\frac{1}{K}(U_1^t + \cdots + U_K^t), \cdots, \frac{1}{K}(U_1^t + \cdots + U_K^t))$,
$Y^{t+1} = (V_1^t - \frac{1}{K}(V_1^t + \cdots + V_K^t), \cdots, V_K^t - \frac{1}{K}(V_1^t + \cdots + V_K^t))$.
$t \leftarrow t + 1$

2.6 Numerical simulation

The algorithms have been coded in C++ and implemented on a Intel Core 2 CPU 2.53 Ghz, RAM 2GB. The directed graph $G = (V, E)$ is randomly generated with m-vertices and n-arcs. The number of QoS constraints (resp. the number of destinations) is p (resp. ℓ).

2.6.1 Unicast QoS Routing

In Unicast QoS routing, we apply the algorithm for solving Problem 2 and the numerical experiment is presented respectively. The datasets are randomly generated in a similar way as the data used in [Chen and Nahrstedt, 1998]. We consider two sets of data. In the first dataset, the weight $w(i, j)$ $((i, j) \in E)$ are random values in the range $[0, 10]$ and the bound constraint L_i are random values in the range $[50, 59]$. In the second dataset, we consider three QoS constraints ($p = 3$): the cost, the time delay, and the packet loss. Denote $w_1(u, v), w_2(u, v), w_3(u, v)$ (resp. L_1, L_2, L_3) are, respectively, the values of cost, the time delay and the packet loss on arc (u, v) (resp. the bound of the cost, the bound of the time delay and the bound of the packet loss). We generate five sub-data sets. Each sub-data set is a random value in a range given below:

	DATA	OBJ-DCA	OBJ-CP12.2	ITE-DCA	T-DCA	T-CP12.2
m=100 n=200 p=10	DATA1	7	7	2	0.12	0.10
	DATA2	9	9	3	0.14	0.11
	DATA3	5	5	3	0.13	0.10
	DATA4	5	4	2	0.12	0.10
	DATA5	6	6	3	0.11	0.09
m=200 n=500 p=20	DATA6	5	5	2	0.55	0.28
	DATA7	7	6	3	0.52	0.25
	DATA8	6	6	2	0.54	0.30
	DATA9	2	2	2	0.54	0.38
	DATA10	xxx	xxx	3	0.53	0.46
	DATA11	8	8	3	0.52	0.30
	DATA12	7	7	2	0.51	0.33
m=1000 n=5000 p=20	DATA13	5	5	2	15.62	12.18
	DATA14	6	6	3	17.11	11.98
	DATA15	2	2	3	14.58	11.13
	DATA16	3	3	3	23.05	14.69
	DATA17	5	4	2	17.61	13.25
	DATA18	5	5	3	19.85	12.71
	DATA19	6	5	3	18.50	12.33
	DATA20	4	4	4	19.89	14.07
	DATA21	7	6	3	17.73	12.38
Average		5.5	5.25	2.67	8.01	5.60

Table 2.1: Comparative results between DCA and Cplex 12.2 in Unicast QoS routing problem.

	DATA	OBJ-DCA	OBJ-CP12.2	ITE-DCA	T-DCA	T-CP12.2
m=500	DATA22	3	3	4	140.33	120.73
n=7000	DATA23	3	2	4	100.26	84.05
p=500	DATA24	2	2	2	83.15	62.50
m=500	DATA25	2	2	3	139.98	107.54
n=10000	DATA26	3	3	3	118.87	86.76
p=500	DATA27	3	3	3	124.27	104.61
m=1000	DATA28	3	3	4	80.67	60.10
n=10000	DATA29	3	3	3	102.02	81.51
p=100	DATA30	3	3	3	145.25	112.12
m=2000	DATA31	5	5	2	150.34	109.86
n=10000	DATA32	5	5	4	149.54	111.66
p=100	DATA33	4	4	3	151.13	115.10
m=2000	DATA34	2	2	4	169.91	131.89
n=12000	DATA35	4	4	3	189.48	157.49
p=100	DATA36	5	5	2	160.98	142.55
m=5000	DATA37	4	4	3	689.06	505.25
n=2000	DATA38	5	5	4	669.07	576.01
p=100	DATA39	6	6	3	702.50	549.96
m=7000	DATA40	6	xxx	3	703.15	xxx
n=25000	DATA41	2	xxx	4	1055.20	xxx
p=100	DATA42	7	7	3	1022.94	698.05
Average		3.81	3.73	3.14	326.09	206.72

Table 2.2: Comparative results between DCA and Cplex 12.2 in Unicast QoS routing problem.

i) sub-data 1: (m,n) = (100,200), $w_1(u,v) \in [0,200]$, $w_2(u,v) \in [0,50ms]$, $w_3(u,v) \in$[0%,5%], $L_1 \in$[600, 660], $L_2 \in [150,165ms]$ and $L_3 \in$[5%,10%].

	DATA	OBJ-DCA	OBJ-CP12.2	ITE-DCA	T-DCA	T-CP12.2
m=100 n=200	DATA43	5	5	3	0.19	0.17
	DATA44	5	5	4	0.18	0.17
	DATA45	4	4	3	0.13	0.11
	DATA46	4	4	2	0.14	0.10
	DATA47	4	4	3	0.13	0.12
m=200 n=400	DATA48	4	4	3	0.57	0.52
	DATA49	3	3	3	0.53	0.36
	DATA50	5	5	4	0.54	0.47
	DATA51	6	6	3	0.40	0.36
m=500 n=1000	DATA52	3	3	4	2.09	1.94
	DATA53	5	5	3	2.20	2.03
	DATA54	6	6	3	2.17	1.96
	DATA55	7	7	3	2.79	2.18
Average		4.69	4.69	3.15	0.93	0.81

Table 2.3: Comparative results between DCA and Cplex 12.2 in Unicast QoS routing problem. ($p = 3$)

	DATA	OBJ-DCA	OBJ-CP12.2	ITE-DCA	T-DCA	T-CP12.2
m=1000 n=2000	DATA56	7	6	3	7.82	6.62
	DATA57	8	8	4	8.04	7.27
	DATA58	9	9	4	12.23	7.51
m=5000 n=7000	DATA59	9	9	3	136.04	127.08
	DATA60	xxx	xxx	3	133.70	125.07
	DATA61	xxx	xxx	4	142.83	132.70
	DATA62	12	12	3	144.97	132.10
	DATA63	16	12	4	152.86	120.94
	DATA64	16	15	4	161.72	129.96
	DATA65	14	14	3	170.59	154.50
	DATA66	10	10	4	151.61	140.19
	DATA67	14	14	4	157.72	133.72
	DATA68	19	19	4	159.34	136.95
Average		12.18	11.64	3.62	118.42	104.20

Table 2.4: Comparative results between DCA and Cplex 12.2 in Unicast QoS routing problem. ($p = 3$)

ii) sub-data 2: the sub-data 1 with the following modifications: (m,n)= (200,400), $w_3(u, v) \in$[0%,2%], $L_3 \in$[1%,5%].

iii) sub-data 3: the sub-data 2 with the following modifications (m,n) = (500,1000), $L_1 \in$[2000,2050], $L_2 \in [500, 515ms]$.

iv) sub-data 4: the sub-data 3 with the following modifications (m,n)= (1000,2000), $L_3 \in$[5%,10%].

v) sub-data 5: the sub-data 4 with the following modifications (m,n)= (5000,7000), $L_1 \in$[7000,7050], $L_2 \in [3000, 3050ms]$, $L_3 \in$[0%,15%].

Table 2.1,2.2 and Table 2.3,2.4 present, respectively, the numerical results for the first dataset and the second dataset. In these tables, OBJ-DCA, OBJ-CP12.2, ITE-DCA, T-DCA, T-CP12.2 and DATA stand for, respectively, the objective value obtained by DCA, and by CPLEX 12.2, the number of iterations of DCA, the CPU time of DCA, the CPU time of CPLEX 12.2 and the name of the generated data (For example, for the "DATA1" (in

	DATA	OBJ-DCA	OBJ-CP12.2	ITE-DCA	T-DCA	T-CP12.2	GAP(%)
	DATA1	22	22	2	0.04	0.04	0.00
m = 14	DATA2	xxx	xxx	3	0.04	0.04	0.00
n = 20	DATA3	19	19	3	0.04	0.04	0.00
p = 3	DATA4	14	14	3	0.04	0.04	0.00
$\ell = 2$	DATA5	13	13	2	0.05	0.07	0.00
	DATA6	15	15	3	0.05	0.05	0.00
	DATA7	47	47	3	0.06	0.05	0.00
m = 50	DATA8	21	21	4	0.11	0.10	0.00
n = 100	DATA9	29	29	4	0.14	0.13	0.00
p = 3	DATA10	16	16	3	0.11	0.12	0.00
$\ell = 2$	DATA11	18	18	3	0.10	0.10	0.00
	DATA12	31	31	3	0.08	0.09	0.00
	DATA57	21	21	3	0.09	0.09	0.00
	DATA58	16	16	3	0.09	0.06	0.00
	DATA59	19	19	4	0.06	0.06	0.00
	DATA60	17	17	3	0.07	0.07	0.00
Average		21.20	21.20		0.080	0.091	0.00

Table 2.5: Comparative results between DCA and Cplex 12.2 in Multicast QoS routing problem.

Table 1), it consists of the generated data for a problem, where $m = 100, n = 200, p = 10$, the values of $w_i(u, v)$ and L_i are given in a file "data1.txt".).

From the numerical results, we observe that:

- DCA always provides an integer solution and it converges after a few number of iterations.
- In the most cases, the objective values given by DCA and CPLEX are the same: 33/41 problems in the first dataset and 21/24 problems in the second dataset (note that, the feasible set of "DATA10" in the first dataset and of "DATA60" and "DATA61" of the second dataset are empty). For the remaining cases the difference is small (one).
- DCA works on all test problems while CPLEX 12.2 fails in some cases.

2.6.2 Multicast QoS Routing

The numerical experiment with respect to Problem 7 is presented following. The datasets are randomly generated in a similar way as the data used in [Chen and Nahrstedt, 1998]. We regard the scenario as the example described in Section I. Herein, three QoS constraints: the travel time, the jitter time delay and the latency traffic jam are considered. In the dataset, the weights $w_1(i, j), w_2(i, j), w_3(i, j), \ ((i, j) \in E)$ (respectively, the travel time, the jitter time delay and the latency traffic jam constraints) are randomly generated in range $[0, 10]$minus, $[0, 10]$s and $[0\%, 1\%]$. The value of $L_1(i), L_2(i), L_3(i)$ are randomly generated in range $[50, 59]$minus, $[50, 60]$s and $[5\%, 6\%]$, respectively. The capacity (resp. the travel cost) of each way are generated in $[1, 5]$ cars (resp. $[1, 10]$).

In these tables (Table 2.5,2.6,2.7,2.8), OBJ-DCA, OBJ-CP12.2, ITE-DCA, T-DCA, T-CP12.2 and DATA stand for, respectively, the objective value obtained by DCA, the one by CPLEX 12.2, the number of iterations of DCA, the CPU time by DCA, the CPU time by CPLEX 12.2 and the name of the generated data (For example, for the "DATA1" (in Table 2.5), it consists of the generated data for a problem, where $m = 14, n = 20, p = 3, \ell = 2$ the values of $w_i(u, v)$, $e(u, v)$, $c(u, v)$ and L_i are given in a file "data1.txt".)

	DATA	OBJ-DCA	OBJ-CP12.2	ITE-DCA	T-DCA	T-CP12.2	GAP(%)
	DATA13	29	29	4	0.09	0.10	0.00
m = 50	DATA14	xxx	xxx	4	0.14	0.14	0.00
n = 100	DATA15	23	23	4	0.08	0.08	0.00
p = 3	DATA16	52	52	3	0.10	0.14	0.00
$\ell = 3$	DATA17	56	56	3	0.11	0.26	0.00
	DATA18	57	57	3	0.11	0.48	0.00
	DATA19	32	32	4	0.12	0.13	0.00
m = 50	DATA20	38	38	4	0.12	0.12	0.00
n = 100	DATA21	70	69	3	0.13	0.12	1.44
p = 3	DATA22	xxx	xxx	3	0.12	0.49	0.00
$\ell = 4$	DATA23	115	108	3	0.22	0.27	6.48
	DATA24	27	27	3	0.11	0.11	0.00
	DATA25	45	45	3	0.10	0.14	0.00
	DATA61	23	23	3	0.10	0.11	0.00
	DATA62	32	31	4	0.11	0.10	3.22
Average		46.077	45.385		0.12	0.14	0.86

Table 2.6: Comparative results between DCA and Cplex 12.2 in Multicast QoS routing problem.

	DATA	OBJ-DCA	OBJ-CP12.2	ITE-DCA	T-DCA	T-CP12.2	GAP(%)
	DATA26	64	62	4	0.16	0.18	3.22
	DATA27	49	48	4	0.20	0.48	2.08
m = 100	DATA28	72	72	3	0.19	0.72	0.00
n = 200	DATA29	19	18	4	0.18	0.17	5.56
p = 3	DATA30	46	46	3	0.19	0.18	0.00
$\ell = 3$	DATA31	44	44	3	0.10	0.11	0.00
	DATA32	53	53	3	0.11	0.58	0.00
	DATA33	24	24	4	0.16	0.16	0.00
	DATA34	95	93	3	0.16	1.10	2.15
	DATA35	xxx	xxx	4	0.16	0.16	0.00
m = 100	DATA36	100	97	4	0.20	0.52	3.09
n = 200	DATA37	65	63	4	0.20	0.32	3.17
p = 3	DATA38	40	40	3	0.11	0.14	0.00
$\ell = 4$	DATA39	xxx	xxx	3	0.10	0.12	0.00
	DATA40	80	78	4	0.13	5.20	2.56
	DATA41	92	91	4	0.16	1.51	1.10
Average		60.214	59.214		0.16	0.73	1.64

Table 2.7: Comparative results between DCA and Cplex 12.2 in Multicast QoS routing problem.

The GAP column presents the value of the gap between OBJ-DCA and OBJ-CP12.2, say:
$GAP = \frac{OBJ_{DCA} - OBJ_{CP12.2}}{OBJ_{CP12.2}}$.
From the numerical result we observe that, like for the unicast routing:

- DCA always provides an integer solution and it converges after a few number of iterations.

- In almost cases, the objective values given by DCA and CPLEX are the same: 16/16 in Table 2.5, 12/15 in Table 2.6, 8/16 in Table 2.7, 9/15 in Table 2.8 (note that the feasible set of "DATA2", "DATA14", "DATA22", "DATA35", "DATA39", "DATA42", "DATA50" are empty). In the rest of DATA, GAP are small. It mean that the objective

	DATA	OBJ-DCA	OBJ-CP12.2	ITE-DCA	T-DCA	T-CP12.2	GAP(%)
	DATA42	xxx	xxx	3	1.80	3.68	0.00
	DATA43	78	78	4	0.38	0.90	0.00
m = 200	DATA44	47	47	3	0.28	0.35	0.00
n = 400	DATA45	114	108	3	0.26	13.72	5.56
p = 3	DATA46	84	81	4	0.36	3.38	3.70
$\ell = 4$	DATA47	112	109	3	0.42	2.64	2.75
	DATA48	70	69	4	0.58	7.23	1.45
	DATA49	92	92	4	0.30	33.62	0.00
	DATA50	xxx	xxx	4	0.18	0.19	0.00
m = 300	DATA51	105	99	4	0.50	35.32	6.06
n = 600	DATA52	98	98	4	0.56	4.28	0.00
p = 3	DATA53	43	43	3	1.16	1.40	0.00
$\ell = 4$	DATA54	142	140	3	0.46	32.31	1.42
	DATA55	101	101	4	0.66	131.51	0.00
	DATA56	97	97	3	0.58	54.16	0.00
Average		91.000	89.385		0.49	24.82	1.61

Table 2.8: Comparative results between DCA and Cplex 12.2 in Multicast QoS routing problem.

value obtained by DCA are rather close to the optimal value (the objective value obtained by CPLEX). 13/17 test instances have GAP which is smaller than 3.70%. 4/17 test instances have GAP between 5.56% and 6.48%.

- In large scale problems, CPU time of DCA is smaller than CPU time of CPLEX. For example, "DATA48", "DATA 49", "DATA52", "DATA54", "DATA55", "DATA56", the GAPs are less than 1.5% while the ratio of CPU time of DCA & CPLEX varies from 10 to 100 times.

2.6.3 Many to many Multicast QoS Routing

The numerical experiment of Problem 9 is presented in the following. The algorithm has been coded in C++ and implemented on a Intel Core 2 CPU 2.8 Ghz, RAM 3GB. The directed graph $G = (V, E)$ is randomly generated with m-vertices and n-arcs. The number of QoS constraints and the number of source-destination sets (resp.) are p and K ($m \in [2300, 2400]$, $n \in [4700, 5000]$, $K \in [90, 100]$, $|D_i| \in [2, 3]$, $c(i, j) \in [10, 39]$, $e(i, j) \in [100, 109]$, $w_i(u, v) \in [1, 10]$, $L_i \in [10000, 10500]$, $\lambda \in [10, 15]$).

In these tables (Tabel 2.9, 2.10), NumVars, NumCons, OBJ-DCA, ITE-DCA, T-DCA and DATA stand for, respectively, number of variables, number of constraints, the objective value obtained by DCA, the number of iterations of DCA, CPU time by DCA and the name of the generated data (For example, for the "D1AM", it consists of the generated data for a problem, where $m = 2400, n = 5000, p = 5, K = 100$ the other values (e.g., $w_i(u, v)$, $e(u, v)$, $c(u, v), \cdots$ and L_i are given in a file "data1AM.txt".)

From the numerical results, we observe that:

- DCA provides an integer solution and converges after a few number of iterations.
- The DCA-PDM is able to solve very large problems, while CPLEX 11.2 cannot solve neither the BILP problem nor Linear relaxation of BILP problem.

Remark: The PDM is a method for solving convex program, so we cannot use PDM directly for solving these problem.

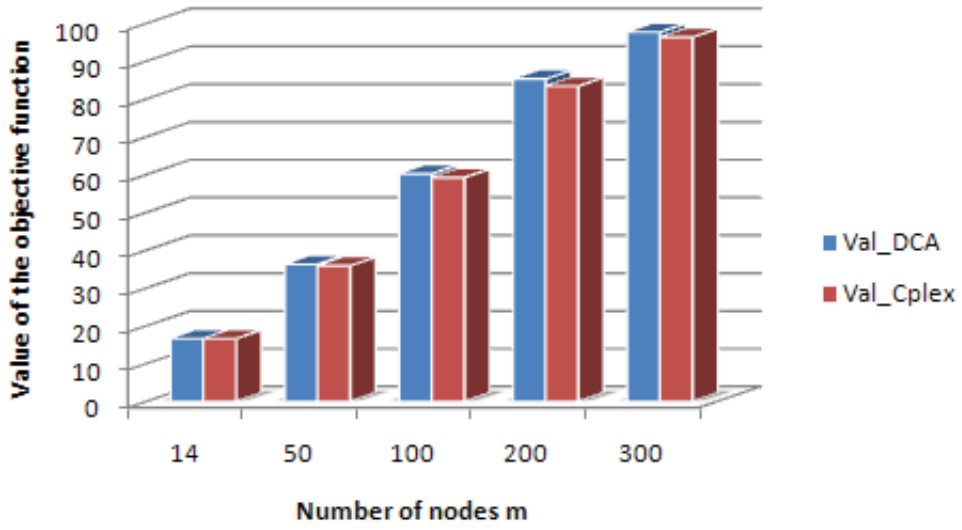

Figure 2.2: The average value of objective functions of DCA and CPLEX 12.2 with m=14, 50, 100, 200, 300 in Multicast QoS Routing problem.

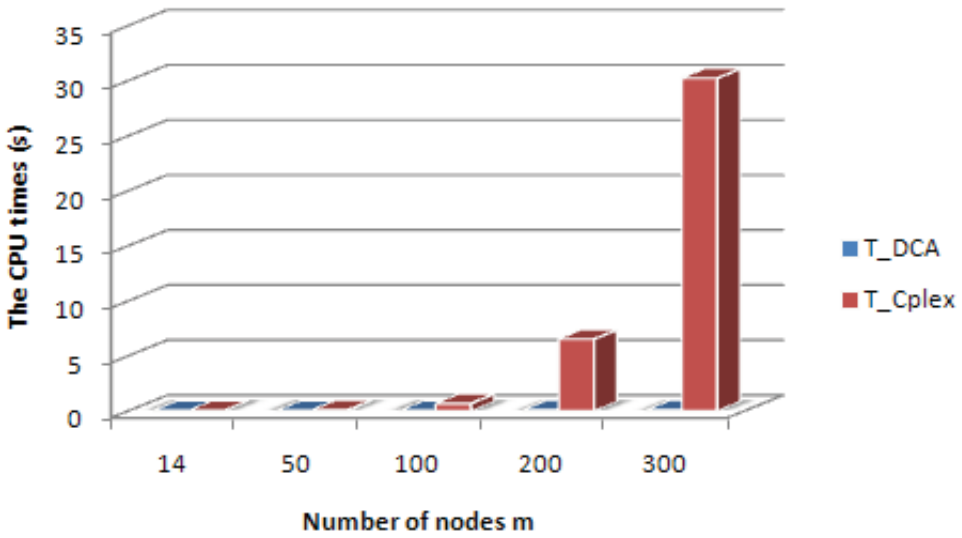

Figure 2.3: The average CPU times of DCA and CPLEX 12.2 with m=14, 50, 100, 200, 300 in Multicast QoS Routing problem.

DATA	NumVars	NumCons	K	OBJ-DCA	ITE-DCA	T-DCA
D1AM	1645000	1520950	100	44147	2	6979.81
D2AM	1635600	1516340	100	49603	2	6779.43
D3AM	1612100	1483530	97	101679	2	6579.22
D4AM	1535000	1426885	90	154494	3	5979.83
D5AM	1590000	1453340	90	154537	2	6127.98
D6AM	1590000	1453340	90	144121	3	6182.94
D7AM	1585000	1458530	90	148958	2	5693.13
D8AM	1585000	1458530	91	150974	2	5601.97
D9AM	1605000	1468150	91	141837	3	5828.78
D1AT	1710800	1553220	100	50096	2	6582.94
D2AT	1461700	1360105	90	217213	3	4895.05
D3AT	1518100	1387765	90	223179	3	5128.89
D4AT	1588600	1457815	95	127108	2	6182.94
D5AT	1555000	1436505	90	146708	2	5306.05
D6AT	1590000	1453340	90	141841	3	5420.33
D7AT	1560000	1438910	90	151135	2	5928.34
D8AT	1585000	1450935	90	144578	3	5976.16
D9AT	1585000	1450935	90	148130	3	5992.30
D10AT	1595000	1463340	91	156572	2	5511.98
D11AT	1590000	1460935	91	147340	3	5731.02

Table 2.9: The numerical simulation results of DCA-PDM algorithm.

DATA	NumVars	NumCons	K	OBJ-DCA	ITE-DCA	T-DCA
D2AQ	1339500	1229225	80	196299	2	4743.97
D3AQ	1447600	1353190	90	194825	2	5727.02
D4AQ	1518100	1409050	93	181798	3	6549.11
D5AQ	1538600	1425620	90	186083	3	6494.14
D6AQ	1590000	1453340	90	154271	3	7442.67
D7AQ	1530000	1424480	90	149301	2	6125.42
D8AQ	1560000	1438910	90	138337	2	6792.95
D9AQ	1550000	1434100	90	144611	2	7321.38
D10AQ	1580000	1448530	90	149159	2	7138.75
D1AY	1536900	1432460	95	149484	2	5484.32
D2AY	1585000	1450935	90	155018	2	5131.36
D3AY	1590000	1453340	90	139431	2	5173.02
D4AY	1595000	1455745	90	149689	2	5103.95
D5AY	1570000	1443720	90	152577	2	5015.41
D6AY	1570000	1451315	91	139672	3	5225.08
D7AY	1560000	1446505	91	146435	3	5170.07
D8AY	1605000	1468150	91	157759	3	5513.13

Table 2.10: The numerical simulation results of DCA-PDM algorithm.

2.7 Conclusion

In this paper, we introduced the CQP and BILP formulation for five Unicast QoS routing problems and two Multicast QoS routing problems. An efficient approach based on DC programming and DCA is proposed for solving these problems. The computational results for Problem 2 and Problem 7 and Problem 9 show that this approach is efficient and original as it gives integer solutions while working in a continuous domain. The DCA is fast and furnished an optimal solution in all the most cases, and a near-optimal solution in the rest cases. For large scale problems we investigated the proximal decomposition technique to solve convex subprograms at each iteration of DCA. Computational results show that this approach is efficient, especially for large-scale settings where the powerful CPLEX fails to be applicable, Our works in progress are devoted to improve DCA-PDM by a more sophisticated implementation, and to develop DCA-Cut to better solve and globally solving these problems.

Chapter 3

Solving Partition Hub Location Routing problem via DCA

This chapter dedicates to analyze Partition Hub Location Routing Problem (PHLRP) and introduce a new mathematical model of PHLRP, then using DC programming and DCA to solve this problem. The main content was considered in [Ta et al., 2012a].

The Partitioning-Hub Location-Routing Problem (PHLRP) is a hub location problem involving graph partitioning and routing features. PHLRP consists of partitioning a given network into sub-networks, locating at least one hub in each sub-network and routing the traffic within the network at minimum cost. There are various important applications of PHLRP, such as in the deployment of network routing protocol problems and in the planning of freight distribution problems. We first present the formulation of this problem as an Binary Integer Linear Programming (BILP) and then investigate a new method based on DC (Difference of Convex functions) programming and DCA (DC Algorithms). Preliminary numerical results are compared with CPLEX, the best solver for BILP. These results show that the proposed algorithm is efficient.

3.1 Introduction

The problem of cost-effectively transferring a set of commodities (e.g., people, goods, data packages, etc) between source-destination pairs is an important problem in the network applications (e.g., in a vehicular network, in a freight distribution network, in a Internet protocol network, etc). In general, a set of specific nodes is created as a *backbone* of transferring throughout the network. The backbone aggregates the traffic flow corresponding to several source-destination pairs. Each node in the backbone is called *a hub* (i.e., transshipment or switching points). The hub location problems consist of finding the backbone in the networks with respect to optimal transfer costs. There are many versions of the hub location problems (see [Alumur and Kara, 2008],[Campbell, 2005]).

In this paper, we study a specific case of the hub location problem called the Partitioning-Hub Location-Routing Problem (PHLRP). The PHLRP consists of partitioning a given network into sub-networks, locating at least one hub in each sub-network, and routing the traffic within the network at minimum cost (see [Ozsoy et al., 2008],[Catanzaro et al., 2011]). We first analyze the formulation of PHLRP in [Ozsoy et al., 2008], and then propose a new formulation in which some constraints are added (see Appendix). For solving the PHLRP with this new formulation, we investigate an efficient approach based on DC programming and DCA. As a result in Chapter 1.3, we show how to adapt this approach for PHLRP formulated as BILP. The proposed DCA for BILP enjoys several advantages: it converges to a local (integer) solution after a finitely many iterations, and requires only the solution of a few number of linear programs. Moreover, it is worth to mention that, although the DCA is a continuous approach working on a continuous domain, it provides an integer solution. This is unusual in continuous approaches and is original property of the proposed DCA. The efficiency of DCA is compared with CPLEX, the best solver for

BILP. The computational results on several test problems show that DCA is an efficient algorithm for solving BILP.

3.2 Problem statement and mathematical model

In this section, we briefly present the notations, the problem statement and a new model of PHLRP, which is obtained by adding the constraints (3.5) into the model of PHLRP (Formulation 3) in [Ozsoy et al., 2008].

3.2.1 Notations and problem statement

- Assume that a network topology is presented as a directed graph $D = (V, A)$. The arc set A is symmetric (i.e., $(j, i) \in A$ for all $(i, j) \in A$).
- Let $(V_1, \cdots, V_k)$ be a partition of V into subsets (i.e., $V_1, \cdots, V_k$ are disjoint subsets of V such that $V = \cup_{i=1}^{k} V_i$).
- The lower and upper bound values on the size of subsets are F_L and F_U, respectively, i.e., $F_L \leq |V_i| \leq F_U$ for $i = 1, 2, \cdots, k$.
- Let $D_i = (V_i, A_i)$, be a sub-network of D induced by V_i, $\quad i = 1, \cdots, k$, respectively.
- A subset H of nodes in V is defined as *hub* nodes.
- Let Y be an upper bound of the number of the hubs which can locate in the network, so we have $|H| \leq Y$. The nodes in $V - H$ are called *local* nodes.
- The sub-network of D induced by H is defined as $D_H = (H, A_H)$, which is called the *backbone*.
- The set of all source-destination pairs which have traffic in-between is a set T. For each pair $(u, v) \in T$, node u is called the *source* and node v is called the *destination*.
- The amount of required traffic flow between source-destination pair $(u, v) \in T$ is defined as $d_{u,v}$.
- When an unit flow associated with the pair $(u, v) \in T$ passes through the arc $(i, j) \in A$, an incurred non-negative cost is $c_{i,j}^{u,v}$.
- A capacity value of the flow on an arc (i, j) is $C_{i,j}$.
- Let $Q \subseteq V$ and $S \subseteq V$. Let $\sigma(Q)$ be the set of all arcs in A whose both end-nodes are in Q, i.e., $\sigma(Q) = \{\{i, j\} \in A : i, j \in Q\}$. The set of arcs in A is denoted by $\sigma(Q, S)$ with one end-node in Q and the other end-node in S, i.e., $\sigma(Q, S) = \{\{i, j\} \in A : i \in Q, j \in S\}$. The set of outgoing arcs from Q and incoming arcs into Q is denoted by $\sigma^+(Q)$ and $\sigma^-(Q)$, respectively, i.e., $\sigma^+(Q) = \sigma(Q, V - Q)$ and $\sigma^-(Q) = \sigma(V - Q, Q)$.
- Let $N_{|V|\times|A|}$ be the node-arc incidence matrix of D, i.e.,

$$N_{u,(i,j)} = \begin{cases} 1 & \text{if } (i,j) \in \sigma^+(\{u\}), \\ -1 & \text{if } (i,j) \in \sigma^-(\{u\}), \\ 0 & \text{otherwise,} \end{cases} \quad \forall u \in V, (i,j) \in A.$$

- Nf is the product of the matrix N by a flow vector $f \in \mathrm{IR}^{|A|}$ and $(Nf)_i$ stands for the i^{th} component of the vector Nf, i.e., $(Nf)_i$ is equal to the net flow for node i. Let $\sum_{i\in S}(Nf)_i$ be $(Nf)_S$.

We assume that routing of traffic between (u, v) pairs in T is in accordance with the routing in ISIS protocol, i.e., the routing in our problem complies with the following assumptions ([Ozsoy et al., 2008]):

$(A1)$ Inter-area traffic flows can be sent through an arc (i, j) only if (i, j) is on the backbone (i.e., $(i, j) \in A_H$).

$(A2)$ Intra-area traffic is realized over paths which entirely lie within the areas.

$(A3)$ Local nodes always use the same hub to send or receive inter-area flows.

Assumptions $(A1)$ and $(A2)$ can be explained in more detail as follows:

– An area can exchange flows with the other areas only if it possesses a hub node (i.e., an area without a hub would be isolated from the rest of the network).

– Hubs of areas with traffic demand in-between have to be connected to each other over the backbone.

– If the source and the destination of a traffic flow are in different areas, then the flow is first directed towards the hub within the source area; subsequently, it is sent over the backbone towards the hub of the destination area; finally, it is directed from this hub towards the destination node.

– If the source and the destination are in the same area, then the flow is simply sent over a path that is entirely contained within the area.

Now, given a network $D = (V, A)$, an informal definition of this problem is represented as:

1. Partitioning D into areas $D_1, D_2, \cdots, D_k$ of size at least F_L and at most F_U (i.e., partitioning V into subsets $V_1, V_2, \cdots, V_k$ such that $F_L \leq |V_i| \leq F_U$ for $i = 1, 2, \cdots, k$),
2. Determining H such that $|H| \leq Y$ (the hub nodes),
3. Assigning every local router to a hub within its area (assumption (A3)),
4. Routing the traffic for every $(u, v) \in T$ in accordance with the assumptions (A1), (A2), and capacity restrictions over the arcs,
5. Minimizing the total cost of routing.

3.2.2 Mathematical model

There are six sets of variables used in this formulation:

- the partitioning variables $w_{u,v} \in \{0, 1\}$ for all $u, v \in V$ such that $u \neq v$ with

$$w_{u,v} = \begin{cases} 1 & \text{if } u \text{ and } v \text{ are in the same area;} \\ 0 & \text{otherwise.} \end{cases}$$

- the hub-location variables $x_{u,v} \in \{0, 1\}$, for all $u, v \in V$, defined as:

$$x_{u,v} = \begin{cases} 1 & \text{if node } u \text{ is assigned to the hub } v \text{ for exchanging flows} \\ & \text{with outside of its area;} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$x_{u,u} = \begin{cases} 1 & \text{if u is a hub;} \\ 0 & \text{otherwise.} \end{cases}$$

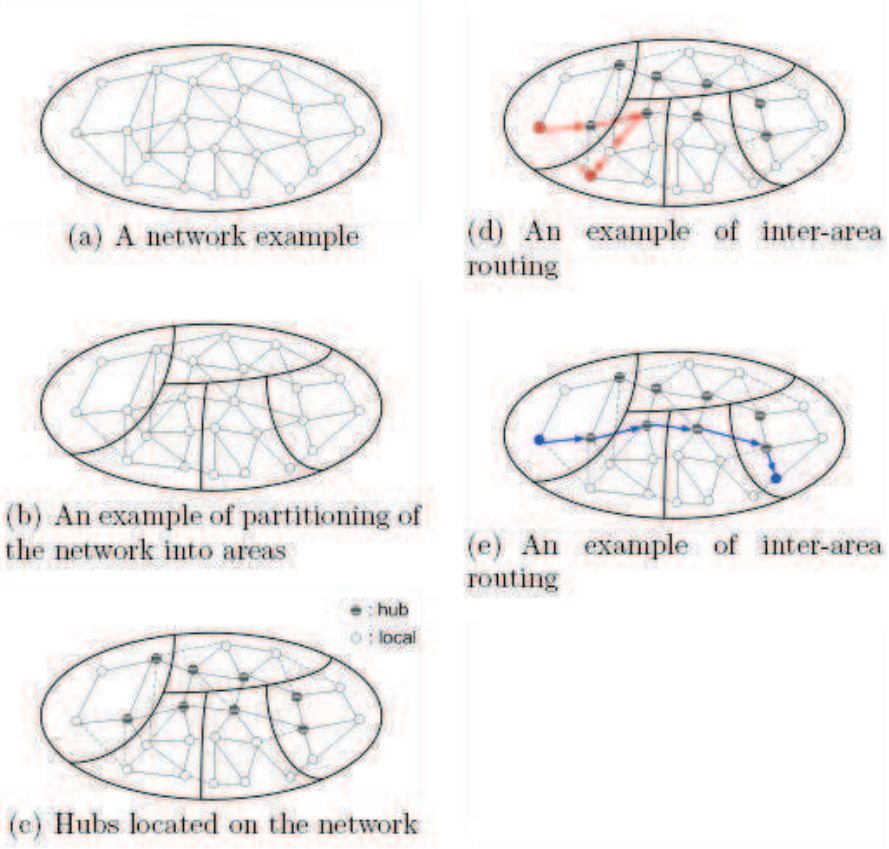

Figure 3.1: An example of partitioning hub location and routing components ([Ozsoy et al., 2008]).

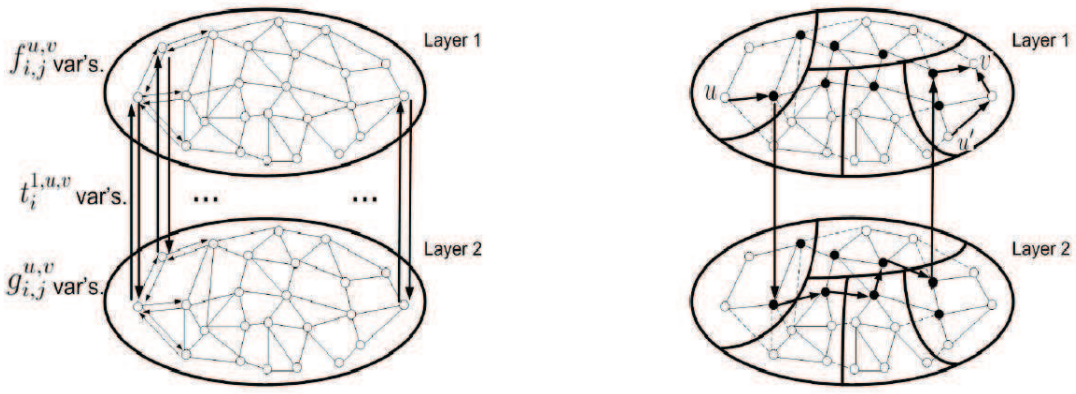

Figure 3.2: An example of partitioning hub location and routing components ([Ozsoy et al., 2008]).

In the formulation below, two replicates of D (see Figure 3.2.1) are using. The definition of costless and incapacitated inter-layer arcs in both directions between identical nodes of the two layers are presented. An inter-layer arc from node i of Layer 1 to node i of Layer 2 is created and vice versa. If a source-destination pair (u, v) lies in the same area, so the flow from u to v is in Layer 1. If u and v lie in different areas, then the flow starts at u of Layer 1, and jumps to Layer 2 over the hub that u is allocated to. It continues within Layer 2 until reaching the hub that v is allocated to, and returns over this hub to Layer 1, where it is sent towards v (see Figure 3.2.1).

In Layer 2, only the flows whose its arcs belong to A_H are permitted. On the other hand, in Layer 1, the flows only through arcs that lie within areas are permitted. Moreover, the inter-layer flows only through the inter-layer arcs corresponding to hubs are permitted.

- $\phi_{i,j}^{u,v}$, $\gamma_{i,j}^{u,v}$ for all $(u, v) \in T$ and $(i, j) \in A$ to represent the flows in Layer 1 and Layer 2, respectively;

$$\phi_{i,j}^{u,v} = \begin{cases} 1 & \text{if the flow from } u \text{ to } v \text{ passes through the arc } (i,j) \text{ in Layer 1;} \\ 0 & \text{otherwise;} \end{cases}$$

$$\gamma_{i,j}^{u,v} = \begin{cases} 1 & \text{if the flow from } u \text{ to } v \text{ passes through the arc } (i,j) \text{ in Layer 2;} \\ 0 & \text{otherwise.} \end{cases}$$

- $\tau_i^{1,u,v}$, $\tau_i^{2,u,v} \in \{0, 1\}$ for all $(u, v) \in T$ and $i \in V$, to represent the inter-layer flows from Layer 1 to Layer 2 and from Layer 2 to Layer 1, respectively;

$$\tau_i^{1,u,v} = \begin{cases} 1 & \text{if the flow from } u \text{ to } v \text{ passes from Layer 1 to Layer 2} \\ & \text{over vertex } i \text{ (i.e., through arc } (i,i) \text{ between two layers);} \\ 0 & \text{otherwise;} \end{cases}$$

$$\tau_i^{2,u,v} = \begin{cases} 1 & \text{if the flow from } u \text{ to } v \text{ passes from Layer 2 to Layer 1} \\ & \text{over vertex } i \text{ (i.e., through arc } (i,i) \text{ between two layers);} \\ 0 & \text{otherwise.} \end{cases}$$

The (PHLRP) problem can be formulated as follows:

$$\min \sum_{(u,v)\in T} \sum_{(i,j)\in A} c_{i,j}^{u,v} d_{u,v} (\phi_{i,j}^{u,v} + \gamma_{i,j}^{u,v}) \tag{3.1}$$

subject to

$$w_{u,v} + w_{u,t} - w_{v,t} \leq 1 \ \ \forall u, v, t \in V : u \neq v, u \neq t, v < t, \tag{3.2}$$

$$\sum_{v \in V - \{u\}} w_{u,v} \leq F_U - 1 \ \ \forall u \in V, \tag{3.3}$$

$$\sum_{v \in V - \{u\}} w_{u,v} \geq F_L - 1 \ \ \forall u \in V, \tag{3.4}$$

$$w_{u,v} = w_{v,u} \ \ \forall u, v \in V, \tag{3.5}$$

$$\sum_{v \in V} x_{u,v} = 1 \ \ \forall u \in V, \tag{3.6}$$

$$x_{u,v} \leq x_{v,v} \ \ \forall u, v \in V : \ u \neq v, \tag{3.7}$$

$$\sum_{u\in V} x_{u,u} \leq Y, \tag{3.8}$$

$$x_{u,v} + x_{v,u} \leq w_{u,v} \ \ \forall u,v \in V : u < v, \tag{3.9}$$

$$\sum_{j\in V,(i,j)\in A} \gamma_{i,j}^{u,v} \leq x_{i,i} \ \ \forall (u,v) \in T, \ \forall i \in V, \tag{3.10}$$

$$\phi_{i,j}^{u,v} + \phi_{j,i}^{u,v} \leq w_{i,j} \ \ \forall (u,v) \in T, \ \forall (i,j) \in A, \ i < j, \tag{3.11}$$

$$\sum_{(u,v)\in T} d_{u,v}(\phi_{i,j}^{u,v} + \gamma_{i,j}^{u,v}) \leq C_{i,j} \ \ \forall (i,j) \in A, \tag{3.12}$$

$$(N\phi^{u,v})_i + \tau_i^{1,u,v} - \tau_i^{2,u,v} = \begin{cases} 1 & \text{if } i = u \\ 0 & \text{if } i \in V - \{u,v\} \\ -1 & \text{if } i = v \end{cases} \quad \forall (u,v) \in T, \tag{3.13}$$

$$(N\gamma^{u,v})_i - \tau_i^{1,u,v} + \tau_i^{2,u,v} = 0 \ \ \forall (u,v) \in T, \tag{3.14}$$

$$\sum_{i\in V} \tau_i^{1,u,v} = 1 - w_{u,v} \ \ \forall (u,v) \in T, \tag{3.15}$$

$$\tau_i^{1,u,v} \leq x_{u,i} \qquad \forall (u,v) \in T, \forall i \in V, \tag{3.16}$$

$$\tau_i^{2,u,v} \leq x_{v,i} \qquad \forall (u,v) \in T, \forall i \in V, \tag{3.17}$$

$$\phi_{i,j}^{u,v}, \gamma_{i,j}^{u,v} \in \{0,1\} \ \ \forall (i,j) \in A, \ (u,v) \in T, \tag{3.18}$$

$$w_{i,j} \in \{0,1\} \ \ \forall i,j \in V, \ i \neq j, \tag{3.19}$$

$$x_{i,j} \in \{0,1\} \ \ \forall i,j \in V, \tag{3.20}$$

$$\tau_i^{1,u,v}, \tau_i^{2,u,v} \in \{0,1\} \ \ \forall i \in V, \ (u,v) \in T. \tag{3.21}$$

The objective function (3.1) presents the total cost of the routing in the network. Constraints (3.2), called triangle inequalities, are ensured the valid partitioning of the network into areas. They were used firstly by M. Grotschel and Y. Wakabayashi in [Grotschel and Wakabayashi, 1990]. These constraints guarantee that if two edges of a triangle lie within an area then the third edge lies within this area too. Constraints (3.3) and (3.4) present upper and lower bounds, F_U and F_L respectively. *Constraints (3.5) are necessary for the definition of* $w_{u,v}$*, because these constraints in [Ozsoy et al., 2008] do not guarantee* $w_{u,v} = w_{v,u}$ $\forall u,v \in V$ *(see Appendix 1).* Constraints (3.6) guarantee that each node is allocated to a hub to communicate with nodes outside its area. Constraints (3.7) check a local node from being assigned to a local node. An upper bound Y of the number of hubs can be located in the network presented by constraints (3.8). Constraints (3.9) guarantee that each local node is assigned to a hub which belongs to the same areas as itself. Constraints (3.10) prevent that the backbone consists of only hubs. In addition, in (3.10) if a node is not a hub then it does not receive any flow from other nodes in Layer 2. Constraints (3.11) guarantee that the flow in Layer 1 do not cut area borders. Constraints (3.12) stand for the restriction of the capacity over the arcs. Constraints (3.13) (resp. (3.14)) are flow balance equations for Layer 1 (resp. Layer 2). Constraints (3.15) ensure that the flow stays within in Layer 1, if the source-destination pair $(u,v) \in T$ lies within in the same area (i.e., if $w_{u,v} = 1$). Constraints (3.16) force to get on the backbone (i.e., to pass to Layer 2) through the hub the source is allocated to (i.e., through $i \in V$ for which $x_{u,i} = 1$). Constraints (3.17) fulfill the symmetrical requirement, that is, the flow is force to get off the backbone on the hub that the destination is allocated to.

The assumptions $(A1) - (A3)$ are complied in this formulation. Constraints (3.10) and (3.11) present the assumption $(A1)$. Constraints (3.11) and (3.13) describe the assumption $(A2)$. Finally, the constraints (3.12) stand for the assumption $(A3)$.

An example of applications

The previous optimization problem may be applied to a practical scenario related to the transport of passengers. To this effect we shall consider a carpooling use case where a community of persons chooses to share the car they own with other passengers going the same way. It is most useful for frequent travelers, for instance those do the same trip everyday between their home and their working place. In conventional carpooling systems, there exists a limited number of rendezvous points where carpoolers can meet and decide on an ad-hoc basis the cars occupation and planning. The lack of flexibility of this kind of solution leads to a bad overall acceptance factor for carpooling as an efficient transport mode.

We consider a dynamic carpooling scenario that will use the following hypothesis:

- There are no predetermined meeting points and these meeting points have the same role than hubs in our algorithm.
- Passengers and drivers planning are computed so that the resulting routes have an optimal cost.

The problem is to optimal at the same time: the positioning of the hubs depending of the starting and arriving points of the travelers, and the routing of the passengers.

3.3 Solving Partitioning Hub Location Routing problem by DCA

In this section, we show how to use DCA for solving the Partitioning Hub Location Routing Problem (PHLRP). By using an exact penalty result, we can reformulate the PHLRP in the form of a concave minimization program. The exact penalty technique aims at transforming the original problem (PHLRP) into a more tractable equivalent DC program. Firstly, we define vectors $\phi, \gamma, w, x, \tau^{1,u,v}, \tau^{2,u,v}$ as follows:

- $\phi = (\phi_{i,j}^{u,v})_{(i,j)\in A,\ (u,v)\in T}$,
- $\gamma = (\gamma_{i,j}^{u,v})_{(i,j)\in A,\ (u,v)\in T}$,
- $w = (w_{i,j})_{i,j\in V,\ i\neq j}$,
- $x = (x_{i,j})_{i,j\in V}$,
- $\tau^{1,u,v} = (\tau_i^{1,u,v})_{i\in V,\ (u,v)\in T}$,
- $\tau^{2,u,v} = (\tau_i^{2,u,v})_{i\in V,\ (u,v)\in T}$.

Let us set $z = (\phi, \gamma, w, x, \tau^{1,u,v}, \tau^{2,u,v})$, a matrix $\widetilde{A}$ and a vector $\widetilde{b}$ are defined such that the constraints (3.2) -(3.17) can be expressed as $\widetilde{A}z \leq \widetilde{b}$.

Let

$$K := \{z \in \mathbb{R}^N : \widetilde{A}z \leq \widetilde{b},\ z \in [0,1]^N\}.$$

The feasible set of (PHLRP) is then

$$S = \{z : z \in K,\ z \in \{0,1\}^N\}.$$

Let us consider the function $p : \mathbb{R}^N \to \mathbb{R}$ defined by:

$$p(z) = \sum_{i=1}^{N} z_i(1 - z_i).$$

It is clear that $p(z)$ is concave and finite on K, $p(z) \geq 0 \ \ \forall z \in K$ and that:

$$\big\{z : z \in S\big\} = \big\{z : z \in K, p(z) \leq 0\big\}.$$

Let c be the cost vector of the linear objective function of (3.1), i.e. c is the vector such that the problem (3.1) can be rewritten as:

$$\min\big\{\sum_{i=1}^{N} c_i z_i : z \in K, p(z) \leq 0\big\}.$$

Following the result of penalty theorem (see Theorem 1.6) we have, for a sufficiently large number η ($\eta > \eta_0$), the equivalent concave minimization problem (3.1):

$$\min\{f_\eta(z) := \sum_{i=1}^{N} c_i z_i + \eta p(z) : z \in K\},$$

which is a DC program of the form:

$$\min\big\{g(z) - h(z) : z \in \mathbb{R}^N\big\}, \tag{3.22}$$

where:

$$g(z) = \chi_K(z) \text{ and } h(z) = -f_\eta(z) = -\sum_{i=1}^{N} c_i z_i - \eta p(z).$$

We have successfully transformed an optimization problem with integer variables into its equivalent form with continuous variables. Notice that (3.22) is a polyhedral DC program where g is a polyhedral convex function (i.e., the pointwise supremum of a finite collection of affine functions).

DCA applied to DC program (3.22) consists of computing, at each iteration k, the two sequences $\big\{z^k\big\}$ and $\big\{v^k\big\}$ such that $v^k \in \partial h(z^k)$ and z^{k+1} solves the next linear program of the form (P_k) (see Introduction)

$$\min\{g(z) - \langle z - z^k, v^k\rangle : z \in \mathbb{R}^N\} \Leftrightarrow \min\{-\langle z\ , v^k\rangle :\ z \in K\}. \tag{3.23}$$

The function h is differentiable and a gradient $v^k \in \partial h(z^k)$ can be computed as follows:

$$v^k = \nabla h(z^k) = 2\eta z^k - \eta \mathbf{e} - c, \tag{3.24}$$

where $e = (1, \cdots, 1)^T \in \mathbb{R}^N$ and $c = (c_1, \cdots, c_N)^T$.

The DCA scheme applied to (3.22) can be summarized in **Algorithm 6**.

Algorithm 6 DCA

Initialization:
Choose a initial point z^0, set $k = 0$;
Let $\varepsilon_1, \varepsilon_2$ be sufficiently small positive numbers;
Repeat
Compute v^k via (3.24);
Solve the linear program (3.23) to obtain z^{k+1};
$k \leftarrow k + 1$;
Until either $\|z^{k+1} - z^k\| \leq \varepsilon_1(\|z^k\| + 1)$ or $|f_\eta(z^{k+1}) - f_\eta(z^k)| \leq \varepsilon_2(|f_\eta(z^k)| + 1)$.

Theorem 3.1 *(Convergence properties of Algorithm DCA)*

- *DCA generates the sequence* $\{z^k\}$ *contained in* $V(K)$ *such that the sequence* $\{f_\eta(z^k)\}$ *is decreasing.*
- *The sequence* $\{z^k\}$ *converges to* $z^* \in V(K)$ *after a finite number of iterations.*
- *The point* z^* *is a critical point of Problem (3.22). Moreover if* $z_i^* \neq \frac{1}{2}$ *for all* $i \in \{0, \ldots, N\}$, *then* z^* *is a local solution to (3.22).*
- *For a number* η *sufficiently large, if at iteration* r *we have* $z^r \in \{0,1\}^N$, *then* $z^k \in \{0,1\}^N$ *for all* $k \geq r$.

Proof 3.1 *See Theorem 1.7.*

	DATA	OBJ-DCA	OBJ-CP9.0	ITE-DCA	T-DCA	T-CPLEX	GAP(%)
	DATA1	183	183	2	0,06	0,35	0
m=9	DATA2	559	559	2	0,10	0,09	0
n=26	DATA3	321	321	2	0,06	0,05	0
$\ell = 3$	DATA4	278	278	2	0,09	0,09	0
$F_U = 4$	DATA5	555	555	9	0,33	1,31	0
$F_L = 3$	DATA6	377	377	2	0,06	0,07	0
Y=3	DATA7	549	525	3	0,14	0,56	4,57
	DATA8	723	703	2	0,08	0,94	2,84
	DATA9	226	226	3	0,14	0,12	0
	DATA10	385	385	3	0,11	0,09	0
	DATA11	429	429	2	0,08	0,37	0
m=9	DATA12	267	267	2	0,08	0,32	0
n=46	DATA13	236	236	3	0,19	0,19	0
$\ell = 3$	DATA14	179	179	2	0,11	0,08	0
$F_U = 4$	DATA15	329	318	3	0,17	0,13	3,46
$F_L = 3$	DATA16	402	402	2	0,14	0,14	0
$Y = 3$	DATA17	386	372	3	0,16	0,21	3,76
	DATA18	399	399	2	0,06	0,12	0
	DATA19	280	280	2	0,12	0,12	0
	DATA20	224	224	2	0,13	0,12	0
	DATA21	149	149	2	0,11	0,09	0
	DATA22	258	258	3	0,24	0,21	0
m=9	DATA23	403	403	2	0,09	0,08	0
n=64	DATA24	365	365	2	0,13	0,35	0
$\ell = 3$	DATA25	322	322	2	0,13	0,35	0
$F_U = 4$	DATA26	219	219	2	0,12	0,12	0
$F_L = 3$	DATA27	156	156	2	0,13	0,12	0
$Y = 3$	DATA28	717	717	3	0,14	0,13	0
	DATA29	106	106	2	0,15	0,15	0
	DATA30	210	210	2	0,13	0,14	0
Average		339,73	337,43	2,53	0,13	0,24	0,49

Table 3.1: Comparative results between DCA and Cplex 9.0.

3.4 Numerical experiment

The algorithm has been coded in C++ and implemented on a Intel Core 2 CPU 2.53 Ghz, RAM 2GB. The directed graph $D = (V, A)$ is randomly generated with m-vertices and n-arcs. The number of pair source-destination in the network is ℓ. The datasets are randomly generated in a similar way as the data used in [Ozsoy et al., 2008]. For each size of the parameters $(m, n, \ell, F_U, F_L, Y)$ we consider ten test problems.

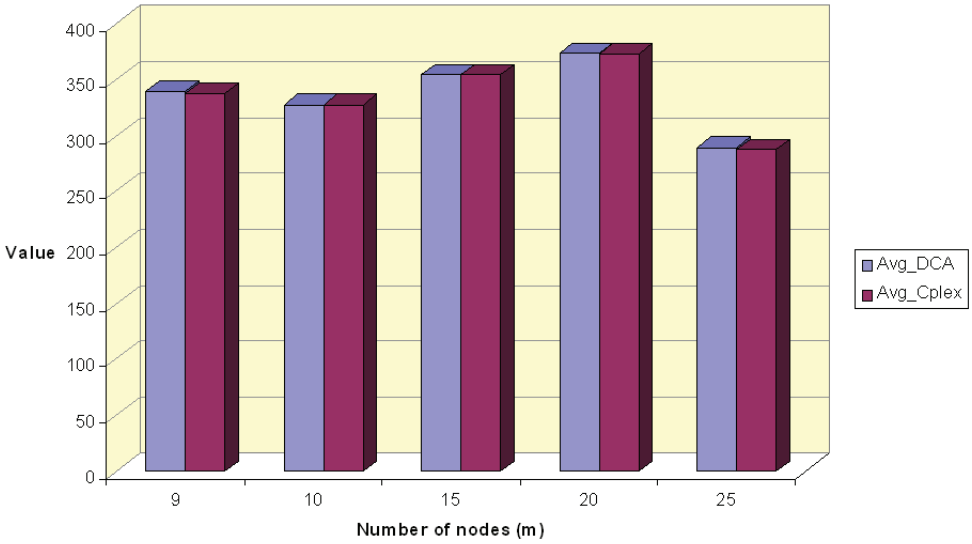

Figure 3.3: Comparison the average value of objective functions between DCA and CPLEX 9.0 with m=9, 10, 15, 20, 25.

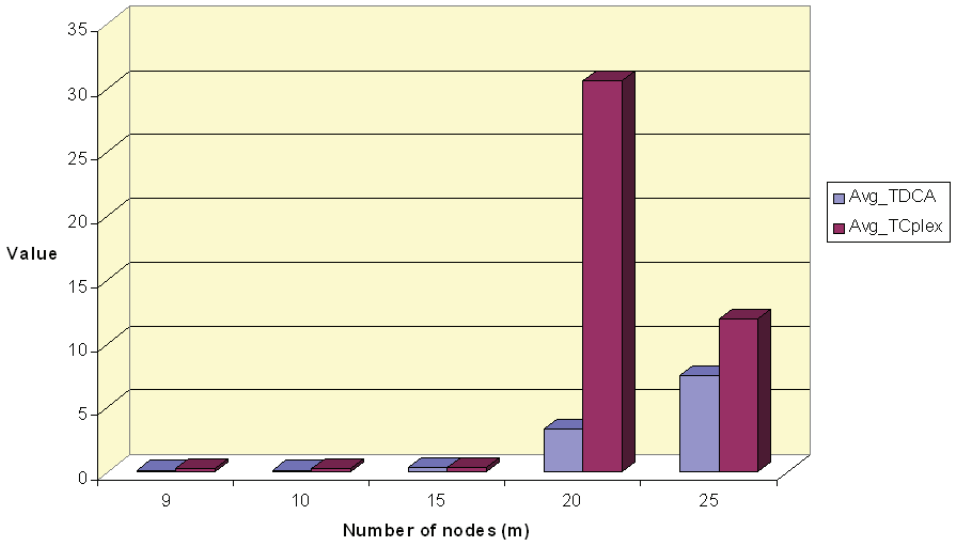

Figure 3.4: Comparison the average value of CPU times between DCA and CPLEX 9.0 with m=9, 10, 15, 20, 25.

	DATA	OBJ-DCA	OBJ-CP9.0	ITE-DCA	T-DCA	T-CPLEX	GAP(%)
	DATA31	407	407	3	0,17	0,15	0
m=10	DATA32	411	411	3	0,18	0,17	0
n=32	DATA33	230	230	2	0,16	0,15	0
$\ell = 3$	DATA34	538	538	2	0,19	0,19	0
$F_U = 4$	DATA35	472	472	2	0,23	0,21	0
$F_L = 3$	DATA36	488	488	3	0,16	0,17	0
Y=3	DATA37	412	412	2	0,18	0,16	0
	DATA38	755	755	2	0,09	0,11	0
	DATA39	358	358	2	0,14	0,12	0
	DATA40	744	744	2	0,11	0,09	0
	DATA41	142	142	2	0,12	0,09	0
m=10	DATA42	125	125	2	0,13	0,12	0
n=46	DATA43	216	216	3	0,19	0,19	0
$\ell = 3$	DATA44	248	248	2	0,11	0,08	0
$F_U = 4$	DATA45	401	401	3	0,09	0,71	0
$F_L = 3$	DATA46	240	240	2	0,14	0,12	0
$Y = 3$	DATA47	246	246	3	0,16	0,25	0
	DATA48	573	573	3	0,16	0,16	0
	DATA49	135	135	2	0,12	0,12	0
	DATA50	126	126	2	0,13	0,14	0
	DATA51	400	400	2	0,09	0,11	0
	DATA52	189	189	2	0,10	0,11	0
m=10	DATA53	246	246	3	0,09	0,11	0
n=72	DATA54	288	288	2	0,13	0,15	0
$\ell = 3$	DATA55	194	194	2	0,10	0,15	0
$F_U = 4$	DATA56	299	299	2	0,12	0,12	0
$F_L = 3$	DATA57	232	232	2	0,13	0,12	0
$Y = 3$	DATA58	100	100	3	0,14	0,13	0
	DATA59	437	437	2	0,08	0,09	0
	DATA60	169	169	2	0,13	0,14	0
Average		327,37	327,37	2,2	0,14	0,16	0

Table 3.2: Comparative results between DCA and Cplex 9.0.

In these tables, OBJ-DCA, OBJ-CP9.0, ITE-DCA, T-DCA, T-CP9.0 and DATA stand for, respectively, the objective value obtained by DCA, the one obtained by CPLEX 9.0, the number of iterations of DCA, the running time of DCA, the running time of CPLEX9.0 and the name of the generated data.

	DATA	OBJ-DCA	OBJ-CP9.0	ITE-DCA	T-DCA	T-CPLEX	GAP(%)
	DATA61	658	658	3	0,22	0,22	0
m=15	DATA62	280	280	2	0,18	0,19	0
n=72	DATA63	433	433	3	0,25	0,25	0
$\ell = 3$	DATA64	510	510	3	0,31	0,32	0
$F_U = 5$	DATA65	509	509	3	0,55	0,49	0
$F_L = 3$	DATA66	619	619	3	0,31	0,61	0
Y=5	DATA67	307	307	2	0,44	0,44	0
	DATA68	372	372	2	0,33	0,32	0
	DATA69	246	246	2	0,43	0,43	0
	DATA70	605	605	2	0,41	0,51	0
	DATA71	180	180	2	0,15	0,11	0
m=15	DATA72	188	188	4	0,65	0,67	0
n=120	DATA73	398	398	3	0,21	0,21	0
$\ell = 3$	DATA74	217	217	2	0,23	0,23	0
$F_U = 5$	DATA75	624	624	3	0,29	0,31	0
$F_L = 3$	DATA76	210	210	2	0,24	0,22	0
$Y = 5$	DATA77	124	124	2	0,26	0,25	0
	DATA78	171	171	2	0,21	0,21	0
	DATA79	231	231	4	0,41	0,42	0
	DATA80	216	216	3	0,51	0,51	0
	DATA81	328	328	3	1,99	2,15	0
	DATA82	375	375	2	7,40	7,54	0
m=20	DATA83	324	324	3	2,09	2,16	0
n=126	DATA84	122	122	4	2,13	2,61	0
$\ell = 3$	DATA85	186	186	4	2,10	2,15	0
$F_U = 5$	DATA86	213	213	4	1,12	1,31	0
$F_L = 3$	DATA87	222	222	3	3,13	3,03	0
$Y = 5$	DATA88	224	224	5	3,14	2,39	0
	DATA89	160	160	3	2,13	2,49	0
	DATA90	410	410	2	1,53	1,94	0
Average		322,07	322,07	2,7	1,11	1,16	0

Table 3.3: Comparative results between DCA and Cplex 9.0.

The GAP column presents the value of the gap between OBJ-DCA and OBJ-CP9.0:

$$GAP = \frac{OBJ_{DCA} - OBJ_{CP9.0}}{OBJ_{CP9.0}}.$$

From the numerical results, we observe that:

- DCA always provides an integer solution and it converges after a few number of iterations.
- In most of cases, the objective values given by DCA and CPLEX are the same: 26/30 in Table 3.1, 30/30 in Table 3.2, 30/30 in Table 3.3, 18/20 in Table 3.4, 20/22 in Table 3.5.
- In the rest of DATA (8 instances), GAPs are small. It means that the objective value obtained by DCA are rather close to the optimal value (the objective value obtained by CPLEX). In all experiments GAP is not larger than 4.57%.
- In general DCA is faster than CPLEX: the average CPU time of DCA is smaller than the one of CPLEX.

	DATA	OBJ-DCA	OBJ-CP9.0	ITE-DCA	T-DCA	T-CPLEX	GAP(%)
	DATA91	856	856	2	2,72	2,72	0
m=20	DATA92	540	540	4	6,83	9,06	0
n=126	DATA93	746	742	6	2,55	2,51	0,54
$\ell = 5$	DATA94	651	651	2	2,51	2,51	0
$F_U = 5$	DATA95	570	556	3	8,35	8,39	2,52
$F_L = 3$	DATA96	629	614	6	5,94	9,17	0
Y=5	DATA97	389	389	3	2,14	2,14	0
	DATA98	430	430	3	2,33	2,32	0
	DATA99	582	582	2	7,97	10,09	0
	DATA100	242	242	6	3,41	3,48	0
	DATA101	204	204	3	3,15	3,31	0
m=20	DATA102	127	127	2	2,45	2,41	0
n=210	DATA103	265	265	3	2,21	2,24	0
$\ell = 3$	DATA104	376	376	3	2,23	2,23	0
$F_U = 5$	DATA105	26	26	2	2,29	2,31	0
$F_L = 3$	DATA106	151	151	3	2,24	2,14	0
$Y = 5$	DATA107	144	144	4	2,66	2,71	0
	DATA108	265	265	3	2,21	2,55	0
	DATA109	453	453	2	3,31	3,32	0
	DATA110	306	306	5	3,51	3,51	0
Average		397,60	395,95	3,35	3,55	3,95	0,15

Table 3.4: Comparative results between DCA and Cplex 9.0.

	DATA	OBJ-DCA	OBJ-CP9.0	ITE-DCA	T-DCA	T-CPLEX	GAP(%)
	DATA111	554	554	3	3,88	3,59	0
m=20	DATA112	326	326	4	2,83	2,39	0
n=210	DATA113	385	385	3	2,55	2,72	0,0
$\ell = 5$	DATA114	417	417	2	3,21	3,25	0
$F_U = 5$	DATA115	425	425	8	2,35	2,78	0
$F_L = 3$	DATA116	621	617	3	5,33	1082,31	0,65
Y=5	DATA117	398	398	2	2,34	2,33	0
	DATA118	501	501	5	3,33	3,53	0
	DATA119	403	403	3	4,25	11,33	0
	DATA120	405	405	3	3,41	3,42	0
m=25	DATA121	186	186	3	2,15	2,42	0
n=320	DATA122	289	289	3	3,45	3,80	0
$\ell = 3$	DATA123	315	315	3	4,21	4,24	0
$F_U = 5$	DATA124	286	286	2	2,23	2,32	0
$F_L = 3$	DATA125	122	122	2	2,39	2,35	0
$Y = 7$	DATA126	516	499	2	2,84	2,84	1,40
$m = 25$	DATA127	422	422	4	8,51	12,47	0
n=320	DATA128	225	225	4	12,21	23,05	0
$\ell = 5$	DATA129	315	315	4	14,31	24,75	0
$F_U = 5$	DATA130	260	260	3	13,51	23,86	0
$F_L = 3$	DATA131	386	386	4	12,45	23,43	0
$Y = 7$	DATA132	152	152	3	11,75	17,71	0
Average		359,50	358,55	3,36	5,61	57,31	0,09

Table 3.5: Comparative results between DCA and Cplex 9.0.

3.5 Conclusion

In this paper, the BILP formulation for Partitioning-hub Location-routing problems given in [Ozsoy et al., 2008] has been improved. An efficient approach based on DC programming and DCA is proposed for solving this problem. The computational results show that

this approach is efficient and original as it can give integer solutions while working in a continuous domain. In a future work we plan to combine DCA and Branch-and-Bound or Branch-and-Cut Algorithm for globally solving these problems.

Part III

OPTIMAL SPECTRUM BALANCING IN DSL NETWORK AND POWER CONTROL IN WIRELESS NETWORK

Chapter 4

Power Control in Wireless Networks using DCA

This chapter devotes to solve Power control problems in wireless networks by DCA. It is well known as a nonconvex optimization problem and difficult to solve. In this chapter, we provides an appropriate DC decomposition and then DCA can work efficiency associated with this decomposition.

Nowadays the wireless networks are becoming an innovative business environment in which new values can be created by competing as well as collaborating enterprises through innovation. Power control is typically used in wireless cellular networks in order to optimize the transmission subject to quality of service (QoS) constraints. It has been shown earlier that the power control problem in the wireless cellular network framework can be efficiently solved using the so-called *geometric programming*. However, in order to enable the application of geometric programming to solve the throughout maximization or weighted sum of data rates maximization problems, SINR + 1 has to be approximated as SINR where SINR is the signal to interference-and-noise ratio. Such change of the original problem formulation is obviously imprecise and might be very loose, especially at low SINR regime which is an usual scenario for systems with CDMA applications. In this paper, based on *difference of convex functions* (DC) programming and *DC Algorithm* (DCA), we investigate a new solution method for solving the aforementioned problems. Albeit sub-optimal, the numerical simulations are compared with *geometric programming* show that the proposed algorithm is an efficient approach.

4.1 Introduction

The technology and business of wireless communication system have been developed dramatically since 1990s. With new mobile satellites coming on line, business arrangements, technology and spectrum allocations make it possible for people to make and receive telephone calls anytime anywhere. Today, the mobile telephone success story calls the wireless communications community to turn its attention to other information services, most of them in the category of "wireless data" communications. Wireless technology is a truly revolutionary paradigm shift, enabling multimedia communications between people and devices from any location. It also underpins exciting applications such as sensor networks, smart homes, telemedicine, and automated highways. Wireless networks continue to develop, usage has grown in 2010. Cellular phones are part of everyday wireless networks, allowing easy personal communications. Inter-continental network systems use radio satellites to communicate across the world. Emergency services such as the police utilize wireless networks to communicate effectively. Individuals and businesses use wireless networks to send and share data rapidly, whether it be in a small office building or across the world. Nowadays the Wireless networks are becoming an innovative business environment in which new values can be created by competing as well as collaborating enterprises through innovation. Wireless networks significantly contribute to the seamless collaboration and interoperability within and outside the enterprises.

In practice, one of the most important problem is *the resources management problem with embedded communication capabilities* (power control, channel assignment, and handoffs). Resource management will be critical problem in the systems that include high speed data applications such as high speed delivery of multimedia information integrated with voice, image and data. Therefore, there is an increased urgency to develop and investigate advanced in radio resource management problem on wireless data transmission.

One important issue of the radio resource management is power control. Power control and resource allocation techniques for cellular communication systems have been a recent focus of intensive studies [Chiang, 2005],[Foschini and Miljanic, 1993],[Yates, 1995]. It has been proposed to use the user signal to interference plus noise ratio (SINR) to adjust the transmitted power [Farrokhi et al., 1998]. To specify in CDMA systems, it is particularly important, where users transmit at the same time over the same frequency bands and their spreading codes are not perfectly orthogonal. Transmit power control is often used to tackle this problem of signal interference [Chiang et al., 2008]. In this way, power control is used to control interference, and therefore, to control also individual user's quality of services (QoS). Various objectives have been considered for developing power control algorithms. Particularly, one can maximize the minimum SINR, minimize total transmitted power, or minimize outage probability in a cellular network [Biguesh et al., 2004], [Chiang, 2005],[Julian et al., 2002], [Farrokhi et al., 1998]. The objective represents a systemwide goal to be optimized; however, individual users' QoS requirements also need to be satisfied. Any power allocation must be satisfied by these minimum requirements constraints from the users. For example, a constrained optimization captures the tradeoff between user-centric constraints and some network-centric objective. Because a higher power level from one transmitter increases the interference levels at other receivers, there may not be any feasible power allocation to satisfy the requirements from all the users. Sometimes there exist a solution which can be satisfied the set of requirements, but when a new user is admitted into the system, the solution is no more feasible power control requirements, or the maximized objective is reduced due to the tightening of the constraint set, leading to the need for admission control and admission pricing, respectively (see [Chiang, 2006b]). Although various iterative methods have been developed to solve the power control problem in cellular wireless systems, these methods are not general to allow a diverse set of QoS constraints and objective functions.

A general framework for the power control based on *geometric programming* was developed in [Chiang, 2005],[Kandukuri and Boyd, 2002]. However, in order to enable the application of geometric programming to solve the throughout maximization or weighted sum of data rates maximization problems, SINR + 1 has to be approximated as SINR. Unfortunately, such approximation might be very imprecise and loose, especially at low SINR regime which is a usual scenario for systems with CDMA applications. Moreover, note that the aggregate system throughput or sum of log(1+SINR), is the actual target for network optimization. It turns out that the resulting problem is NP-hard. Although the original formulations can be solved using the method of successive convex-approximation as described in [Chiang, 2005], the number of iterations, i.e., the number of geometric programs that need to be solved, may be large, and thus, leads to high overall complexity. Moreover, global optimality is also not guaranteed.

QoS provisioning in a wireless network is a particularly difficult task because physical layer problems; such as path loss, fading, and multi-path; can make the communication links unreliable. Since the channel gains vary all the time, there is a need for low-complexity algorithms to carry out resource allocation, i.e., power control, in wireless networks. Moreover, such algorithms should also provide good performance in order not to waste radio resources. This is precisely our aim in this work. Specifically, in this paper, we directly consider the aggregate system throughput as objective function in solving the power con-

trol problem for cellular systems. We show that the corresponding optimization problem belongs to the class of so-called DC programming problems which can be globally and efficiently solved using modern optimization methods [Horst et al., 1995]. However, due to the high complexity of global DC programming, we propose low complexity *DC Algorithms* (DCA) to solve the resulting problems. Albeit sub-optimal, the proposed approach achieves *near optimal* results.

In this paper we show how to use this approach for solving the aforemention problem. The numerical simulations are compared with *geometric programming* show that the proposed DCA is an efficient algorithm.

4.2 System model

We consider a cellular network with K users (links[1]) and a single base station. Uplink transmission is considered in this paper although a more general system set up, such as an ad hoc network, is also applicable. An extension to multiple base stations is straightforward.

The effects of three signal strength attenuation factors: path loss, shadowing, and multipath fading are considered in the following propagation model. Let P_k be the transmitted power level of the k^{th} user. Then, the propagation model for the k^{th} user can be written as [Chiang, 2005]

$$\tilde{P}_k = P_k F_k \left(\frac{d_0}{d_k}\right)^{\beta_k}, \tag{4.1}$$

where $\tilde{P}_k$ is the received power, d_k is the propagation path length, d_0 is a reference distance for the antenna far-field, F_k is multipath fading gain, and β_k is the path loss exponent for the k^{th} user. Note that in the aforementioned model we ignore the effect of shadowing for brevity. Using (4.1), the SINR for the k^{th} receiver can be defined by

$$\mathrm{SINR}_k = \frac{L P_k F_k d_0^{\beta_k} d_k^{-\beta_k}}{\sum_{j\neq k}^{K} P_j F_j d_0^{\beta_j} d_j^{-\beta_j} + \sigma_k^2}, \tag{4.2}$$

where σ_k^2 is the noise power at the common base station and $L := W/R > 1$ is the spreading gain of the CDMA system, where W is the chip rate and R is the data rate of the user.

Although SINR is often used as a QoS parameter, it is the network throughput which is of concern. Indeed, it is well known that the capacity of Gaussian channel with Gaussian interference is a function of SINR. Then the throughput for the k^{th} user is given by

$$\mathrm{R}_k = \log_2(1 + \mathrm{SINR}_k). \tag{4.3}$$

Let us denote $\bar{G}_j = F_j d_0^{\beta_j} d_j^{-\beta_j} > 0$, $\forall j$ for the path attenuation of interfering user j. One popular power control problem is based on maximizing the weighted sum of data rates under the peak power constraints for all users. Mathematically, this problem can then be formulated as follows.

Problem 1:

$$\text{maximize} \quad \sum_{k=1}^{K} w_k \mathrm{R}_k \tag{4.4}$$

$$\text{subject to} \quad 0 \leq \sum P_k \leq P^{\mathrm{UB}}, \tag{4.5}$$

$$0 \leq P_k \leq P_k^{\max} \ \ \forall k, \tag{4.6}$$

[1] Each link represents a unidirectional path from the transmitter to the receiver.

where w_k, $k = 1, \ldots, K$ are weights, P^{UB} is the upper bound of the total transmitted power. Note that constraint (4.6) limits the transmitted power of each transmitter to be smaller than the available power $P_k^{\max}$, $\forall k$.

The aforementioned power control scheme in wireless cellular networks takes into account the fairness consideration since the fairness among different users is also a major issue in a QoS policy. In other words, fairness issues must also be taken into account for low priority users. This weight w_k, $\forall k$ reflects this fairness.

Problem 1 is clearly a nonlinear nonconvex optimization problem which is extremely hard to solve. The goal of the following discussion is to show that it can be rewritten in the form of the so-called *DC programming* problem.

4.3 Power control via DC Programming

4.3.1 DC Algorithms for the Proposed Power Control problem

First, we rewrite Problem 1 in the form of a minimization program

Problem 2:

$$\text{minimize} \quad -\sum_{k=1}^{K} w_k R_k \tag{4.7}$$

$$\text{subject to} \quad 0 \leq \sum P_k \leq P^{UB}, \tag{4.8}$$

$$0 \leq P_k \leq P_k^{\max}, \forall k. \tag{4.9}$$

Denote $P = (P_1, \cdots, P_K)^T$. The feasible set C of Problem 2 (4.7)-(4.9) is

$$C := \{P \in \mathbb{R}^K | \sum_{k=1}^{K} P_k \leq P^{UB},\ 0 \leq P_k \leq P_k^{\max},\ k = \overline{1, K}\}.$$

Obviously, C is a polyhedra convex set in $\mathbb{R}^K$.

Note that, we do not impose the constraints on the minimum data rate for each user. However, if there are such constraints, they can also be expressed as linear constraints on $P_1, \ldots, P_k$ as follows

$$\begin{aligned} \mathrm{R}_k \geq \gamma_k^{\mathrm{LB}} &\iff \mathrm{SINR}_k \geq 2^{\gamma_k^{\mathrm{LB}}} - 1 \\ &\iff -LP_k\bar{G}_k + \bar{\gamma}_k\Big(\sum_{j\neq k}^{K} P_j\bar{G}_j + \sigma_k^2\Big) \leq 0 \end{aligned}$$

where $\bar{\gamma}_k = 2^{\gamma_k^{\mathrm{LB}}} - 1$. Thus, all the user's rate constraints are recast as convex linear inequality constraints.

For simplicity of notion, let us set

$$a^k = (\overline{G}_1, \cdots, \overline{G}_{k-1}, L\overline{G}_k, \overline{G}_{k+1}, \cdots, \overline{G}_K)^T,$$

$$b^k = (\overline{G}_1, \cdots, \overline{G}_{k-1}, 0, \overline{G}_{k+1}, \cdots, \overline{G}_K)^T,$$

$$h_k(P_1, \ldots, P_K) = -w_k \log_2\Big(\sum_{j\neq k}^{K} P_j\bar{G}_j + \sigma_k^2\Big), \tag{4.10}$$

$$g_k(P_1, \ldots, P_K) = -w_k \log_2\Big(\sum_{j\neq k}^{K} P_j\bar{G}_j + LP_k\bar{G}_k + \sigma_k^2\Big). \tag{4.11}$$

Thus, we have

$$g_k(P) = -w_k \log_2(\langle a^k, P\rangle + \sigma_k^2),$$
$$h_k(P) = -w_k \log_2(\langle b^k, P\rangle + \sigma_k^2),$$
$$w_k R_k = h_k(P) - g_k(P).$$

The objective function of Problem 2 (4.7)-(4.9) can be rewritten as

$$f(P) = \sum_{k=1}^{K}(g_k(P) - h_k(P)) = g(P) - h(P), \tag{4.12}$$

where $h(P) = \sum h_k(P)$ and $g(P) = \sum g_k(P)$. Since $h_k(P), g_k(P),\ k = 1, \cdots, K$ are convex functions then $h(P)$ and $g(P)$ are convex functions.

However, from numerical points of view, DCA scheme corresponding to this DC decomposition is not interesting because it requires an iterative algorithm for solving a convex program at each iteration. In an elegant way, we introduce a nice DC reformulation of Problem 2 for which the resulting DCA is explicitly determined via a very simple formula. Such a DC decomposition of f is inspired by the following result.

Theorem 4.1 *There exists $\rho > 0$ such that the function $H(P) := \frac{1}{2}\rho||P||^2 - f(P)$ is convex on C.*

Proof 4.1 *We have*

$$\nabla g_k(P) = \frac{w_k}{\log 2}\frac{-a^k}{\langle a^k, P\rangle + \sigma_k^2}\ , \nabla h_k(P) = \frac{w_k}{\log 2}\frac{-b^k}{\langle b^k, P\rangle + \sigma_k^2}\ .$$

Hence,

$$\nabla^2 g_k(P) = \frac{w_k}{\log 2}\frac{a^k(a^k)^T}{(\langle a^k, P\rangle + \sigma_k^2)^2}\ ,\ \nabla^2 h_k(P) = \frac{w_k}{\log 2}\frac{b^k(b^k)^T}{(\langle b^k, P\rangle + \sigma_k^2)^2}\ .$$

Because $g_k(P), h_k(P)$ are convex functions, then $\nabla^2 g_k(P)$ and $\nabla^2 h_k(P)$ are semi-definite positive.
Moreover,

$$\nabla^2 g(P) = \sum_{k=1}^{K}\nabla^2 g_k(P),\ \nabla^2 h(P) = \sum_{k=1}^{K}\nabla^2 h_k(P)$$

are semi-definite positive.
On the other hand, we have

$$||\nabla^2 g_k(P)|| \leq \frac{w_k}{\log 2}\frac{1}{\sigma_k^4}\max_i\{a_i^k\}(\sum_{i=1}^{K} a_i^k).$$

So, if

$$\rho \geq \sum_{k=1}^{K}\frac{w_k}{\log 2}\frac{1}{\sigma_k^4}\max_i\{a_i^k\}(\sum_{i=1}^{K} a_i^k)$$

then

$$\rho I - \sum_{k=1}^{K}\nabla^2 g_k(P) = \rho I - \nabla^2 g(P)$$

is semi-definite positive. Thus, $\frac{1}{2}\rho||P||^2 - g(P)$ is a convex function. Consequently,

$$H(P) = \frac{1}{2}\rho||P||^2 - f(P) = \frac{1}{2}\rho||P||^2 - g(P) + h(P)$$

is a convex function.

Using the theorem above, we get the next DC decomposition of f:

$$G(P) = \frac{1}{2}\rho||P||^2 + \chi_C(x), \quad H(P) = \frac{1}{2}\rho||P||^2 - f(P),$$

and Problem 2 (4.7)-(4.9) can be now written in the standard form of DC program:

$$\min\{G(P) - H(P) \mid P \in \mathbb{R}^K\}. \tag{4.13}$$

4.3.2 DCA applied to Problem 2 (4.7)-(4.9)

Starting with $P^{(0)} \in C$, we have to compute two sequences $\{P^{(k)}\}$ and $\{Q^{(k)}\}$ such that

$$Q^{(k)} \in \partial H(P^{(k)}) \text{ and } Q^{(k+1)} \in \partial(G)^*(Q^{(k)}).$$

As indicated above $P^{(k+1)}$ is an optimal solution of the convex quadratic program

$$\min\{\frac{1}{2}\rho||P||^2 - \langle P, Q^{(k)}\rangle | P \in C\}. \tag{4.14}$$

The DCA applied to Problem 2 can be summarized as **Algorithm 7**.

Algorithm 7

Initialization:

* Choose an initial point $P^{(0)} \in C$, set $r = 0$;
* Let ε be a sufficiently small positive number;

Repeat:

* Calculate

$$Q^{(r)} = \nabla H(P^{(r)}) = \rho\, P^{(r)} - \nabla f(P^{(r)})$$

* Calculate $P^{(r+1)} \in \partial(G)^*(Q^{(r)})$ by solving the linear constrained quadratic program

$$\min\{\frac{1}{2}\rho||P||^2 - \langle P, Q^{(r)}\rangle | P \in C\} \tag{4.15}$$

* Set $r \leftarrow r + 1$;

Until either $||P^{(r+1)} - P^{(r)}|| \leq \varepsilon(||P^{(r)}|| + 1)$ or $|f(P^{(r+1)}) - f(P^{(r)})| \leq \varepsilon(|f(P^{(r)})| + 1)$

Theorem 4.2 (Convergence properties of Algorithm DCA)

i) **Algorithm 7** *generates a sequence* $\{P^k\}$ *such that the sequence* $\{f(P^k)\}$ *is monotonously decreasing.*

ii) The sequence $\{P^k\}$ *converges to the point* P^* *which satisfies the necessary local optimality condition.*

Proof 4.2 *(i) and (ii) are direct consequences of the convergence properties of general DC programs and the fact that* f *is differentiable.*

Complexity analysis:

As we have shown, at each iteration, the algorithm requires solving a convex quadratic program with linear constraints. This convex quadratic program can be solved using any standard method. DCA has a linear convergence for general DC programs.

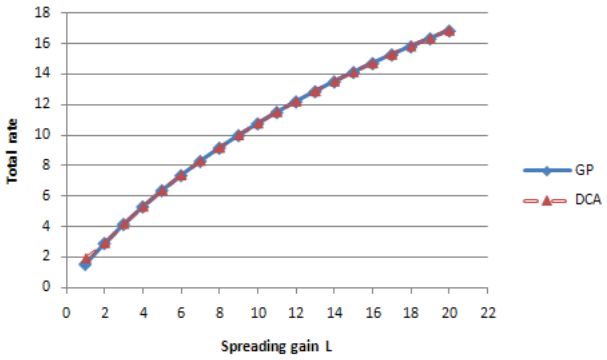

Figure 4.1: Comparative results between DCA and Geometric Programming.

4.3.3 Initial point for DCA

Based on the ideal of SCALE algorithm for solving Optimal Spectral Balancing (OSB) problem in DSL network [Papandriopoulos and Evans, 2006] (see Appendix 3), we propose to use SCALE algorithm applying for Power control problem and finding the initial point for DCA.

In SCALE for solving Power control problem (SCALEP) algorithm, at each iteration, the calculation is quite tractable but the number of iteration of SCALEP is usually large. While the number of DCA usually small and if we have *a good initial point*, DCA converges to *a good solution*. There is a simple ideal that combine SCALEP and DCA algorithms. We can fix a small number of iteration of SCALEP, after that we get the value as initial point for DCA. We can see the improve of this technic in numerical experiment.

4.4 Simulation Results

The algorithm has been coded in Matlab 7.5 and implemented on an Intel Core 2 CPU 2.53 GHz, RAM 2 GB. Our algorithm is compared with Geometric programming (GP) approach, which is a standard method for solving Power control problem [Chiang, 2005], and the GP code is supported in [Grant and Boyd, 2010] with CVX implementation.

We consider a single cell consisting of a base station located at the origin and $K = 10$ users randomly distributed in a two-dimensional region of the size $100m \times 100m$. The distance of the user with medium distance to the base station is used as a reference. The power drop off factor $\beta = \beta_k = 3.7,\ \forall k$. The noise power is $\sigma = \sigma_k^2 = 5mW,\ \forall k$ for all users. We perform 200 fast-scale fading channel realizations for each simulation scenario.

Fig. 4.4 shows the ergodic capacity of the system when the spreading gain changes from 1 to 20 and $P_k^{\max} = 2.0, \forall k$ obtained by DCA and GP. Note that the smaller the spreading gain, the lower SNR regime. Fig. 4.4 presents the average running time of DCA and GP. We can see that, when the spreading gain increased from 1 to 20, the result obtained by DCA is slightly better than the one obtained by GP. However, the average running time of DCA is about 10 times better than the one of GP. Fig. 2 shows the increasing speed of running time of DCA and GP when the number of user increases. We see that, the average running time of GP increases dramatically when the number of users increases while the one of DCA is quite stable. Even though, when the number of users are greater or equal to 30, the GP approach can't give any results, while DCA can solve the problem with 500 or more users.

Table 4.1 and Table 4.2 show more insights into the performance of DCA and GP. In Table 4.1, 4.2, K stands for the number of users, Time-DCA and Time-GP stand for the average running time (in seconds) of DCA and GP, respectively. In Table 4.1 and

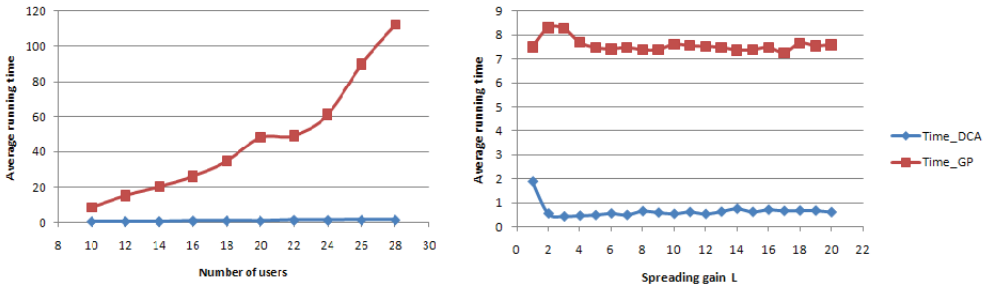

Figure 4.2: Comparative average running time between DCA and Geometric Programming.

Table 4.2, we can see more details about the efficiency of DCA when the number of users increases: the average running time of DCA is always much smaller than the one of GP. Note that GP do not solve the problem when the number of users is greater or equal to 30, while DCA works always, say DCA gives a solution in all cases.

Table 4.1: Comparative results between DCA and Geometric Programming.

K	10	12	14	16	18	20	22	24	26
Time-DCA	0.6251	0.6969	0.6469	0.9531	1.0438	0.9281	1.3501	1.3281	1.4312
Time-GP	8.7565	15.5156	20.3594	26.1875	34.6250	48.3532	49.3874	61.2063	89.7907

Table 4.2: Comparative results between DCA and Geometric Programming.

K	28	30	34	36	40	50	80	200	500
Time-DCA	1.4751	1.6969	1.5188	1.7969	1.9188	2.6375	4.1469	20.6156	159.7063
Time-GP	112.0655	——	——	——	——	——	——	——	——

L	GP	DCA	Time-DCA	Time-GP
1	1.5147	1.8975	1.8887	7.4929
2	2.8859	2.8902	0.5425	8.2988
3	4.1396	4.1424	0.4138	8.2714
4	5.2897	5.2927	0.4456	7.6876
5	6.3578	6.3603	0.4713	7.4669
6	7.3484	7.3514	0.5409	7.4120
7	8.2795	8.2834	0.4766	7.4716
8	9.1498	9.1544	0.6397	7.3729
9	9.9731	9.9768	0.5791	7.3800
10	10.7481	10.7544	0.5238	7.6071
11	11.4922	11.4934	0.6047	7.5484
12	12.1915	12.1961	0.5197	7.5095
13	12.8626	12.8648	0.6184	7.4797
14	13.5079	13.5087	0.7419	7.3597
15	14.1227	14.1232	0.6088	7.3743
16	14.7055	14.7063	0.6991	7.4776
17	15.2792	15.2793	0.6506	7.2482
18	15.8079	15.8113	0.6644	7.6587
19	16.3351	16.3366	0.6628	7.5445
20	16.8475	16.8481	0.5981	7.5920

Table 4.3: Comparative results between DCA and Geometric Programming.

From the numerical results, we observe that:

- The objective function obtained by DCA is always slightly better than by GP.
- The average running time of DCA is much faster than of GP, most of the ratio time (Time-GP/Time-DCA) in range [10,20].
- When the number of users increases, DCA can solve the large-scale problems while GP cannot.

4.5 Conclusion

In this paper, we have developed various QoS provisioning problems for wireless cellular networks based on the resource allocation perspective. The individual user data rate or aggregate system throughput are used as performance metrics. The optimization of the queuing delay is also considered. The consider problem can be solved efficiently by DCA. Numerical simulation example demonstrates the effectiveness and the performance of the proposed approach.

Chapter 5

Optimal Spectrum Balancing in DSL Network using DCA

This chapter presents how to solve Optimal spectrum balancing problem by DCA. From the mathematical point of view, it can be seen as a generic problem of Power control problem in Chapter 4. This problem is solved in a similar way by DCA.

Dynamic spectrum management (DSM) is an effective technique for mitigating detrimental effect of crosstalk in Digital Subscriber Lines (DSL). Among various DSM techniques, centralized Optimal Spectrum Balancing (OSB) achieves the maximum data rates by computing the optimal PSDs (power spectral density) for all modems in DSL systems. In this chapter, we investigate a new and efficient algorithm based on DC programming and DCA for solving nonconvex optimization problems in OSB. Preliminary numerical experiments on real-world data show the efficiency of the proposed algorithm.

5.1 Introduction

Crosstalk is the dominant source of performance degradation in DSL systems, and can severely limit system performance if not mitigated. The effects of crosstalk can be mitigated through spectrum management in an interference-limited DSL system. Traditional static spectrum management (SSM) techniques employ identical spectral masks based on worst-case scenarios for all modems. Consequently, these spectral masks are unduly restrictive and lead to conservative performance. DSM [Song et al., 2002][Report, 2005] recently gains popularity as a new paradigm, aiming to jointly adapt PSDs of each modem based on physical-channel characteristics to minimize crosstalk and to enhance achievable rates.

Distributed Iterative Water Filling (IWF) [Yu et al., 2002], one of the first DSM algorithms, offers significant rate enhancement as compared to SSM techniques. Another algorithm known as Selective Iterative Water Filling (SIW) [Xu et al., 2007] was introduced. SIW selectively applies IWF to users in the under-utilized sections of the frequency spectrum until all users use up their total power. While SIW shows significant performance gains over IWF, the performance is still sub-optimal.

A centralized Optimal Spectrum Balancing (OSB) approach based on dual decomposition with the assumed explicit coordination among the users and hence complete knowledge of all channel responses, was introduced to solve the rate maximization problem, and demonstrated better performance than distributed IWF [Cendrillon et al., 2006]. Its computational complexity is linear in the number of frequency tones , but exponential in the number of users . The exponential part of the complexity in N, i.e., , is due to exhaustive search on each tone to find the optimal power allocation tuple. This exhaustive search essentially makes OSB intractable with more than 5-6 users.

The principal reason for obtaining the optimum solution via exhaustive search is that the optimization problem in OSB is highly non-convex and has many local maxima as argued in [Cendrillon et al., 2006]. Conventional local optimization techniques, which are designed for finding one of local optima, cannot be applied to obtain global maxima for

the non-convex optimization problem in OSB. Hence, [Cendrillon et al., 2006] proposed to use exhaustive search to find the global maxima, which inevitably leads to exponential complexity in N. Another approach [6] attempted to find global optimum by computing all roots of the first-order necessary conditions and boundary points of the optimization problem, and then selecting the one with largest value. However, this method cannot be generalized for multiple users.

Recently, [Cendrillon and Moonen, 2005]-[Cendrillon et al., 2007] proposed several low-complexity heuristic DSM algorithms based on OSB. Among low-complexity heuristic algorithms [Cendrillon and Moonen, 2005]-[Cendrillon et al., 2007], two near-optimal low-complexity centralized algorithms called Iterative Spectrum Balancing (ISB) were introduced in [Cendrillon and Moonen, 2005], [Lui and Yu, 2005], whose main idea is to locally optimize non-convex function in OSB by employing coordinate descent method through a series of line searches. The other two autonomous low-complexity algorithms SCALE and ASB were presented in [Papandriopoulos and Evans, 2006], [Cendrillon et al., 2007], respectively. But none of these low-complexity heuristic algorithms can guarantee finding the global optimum for the non-convex problem in OSB.

It is of key importance to notice that basic components of the objective function of nonconvex optimization problem in OSB are logarithmic bit-loading rate functions of signal to interference and noise ratio (SINR) at receiver side. Since, thus, instead of manipulating the bit-loading rate function as a single nonconvex function, we can express it alternately as the difference of two logarithmic functions. Because the resulting logarithmic functions are both convex, this objective function can be expressed as the difference of two convex functions (each convex function is called DC component of the objective function). In recent years there has been a very active research in DC (Difference of Convex functions) programming. A great deal of work involves global optimization (which is concerned with finding global solutions to nonconvex programs) whose main tools and solution methods are developed according to the spirit of the combinatorial optimization, but with the difference that one works in the continuous framework (see [Horst et al., 1995, Horst and Tuy, 1996]). However, most robust and efficient global algorithms actually do not meet the expected desire: solving real life programs in their true dimension. In contrast with the combinatorial approach where many global algorithms have been studied, there have been a very few algorithms for solving DC programs in the convex analysis approach where the convexity of the two DC components of the objective function has been used to develop appropriate tools from both theoretical and algorithmic viewpoints.

5.2 Spectrum Management problem (SMP)

Discrete multitone (DMT) [Starr et al., 1999] has been adopted as standard in various DSL applications such as asymmetric DSL (ADSL) and more recently for VDSL by ITU (International Telecommunication Union). For a sufficiently large number of sub-carriers, DMT transmission over a frequency-selective fading channel can be modeled as a set of K parallel independent flat-fading sub-carrier AWGN channels. Under this Gaussian assumption, the achievable bit-loading rate of user n on tone k is

$$\begin{aligned} r_k^n &= \log_2(1+\frac{1}{\Gamma}\frac{\left|g_k^{n,n}\right|^2 p_k^n}{\sum\limits_{m\neq n}\left|g_k^{n,m}\right|^2 p_k^m+\omega_k^n}) \\ &= \log_2(1+\frac{1}{\Gamma}\frac{p_k^n}{\sum\limits_{m\neq n} h_k^{n,m} p_k^m+\sigma_k^n}) \end{aligned} \tag{5.1}$$

where

- p_k^n denote user n's transmit PSD (Power spectral density) on tone k.
- ω_k^n denote user n's transmit noise power on tone k.
- $g_k^{n,m}$ is the channel path gain from user m to user n on tone k.
- $h_k^{n,m} = \frac{|g_k^{n,m}|^2}{|g_k^{n,n}|^2}$ is the normalized interference path power gain from user m to user n on tone k.
- $\sigma_k^n = \frac{\omega_k^n}{|g_k^{n,n}|^2}$ is the noise variance of user n on tone k.
- Γ is the SNR-gap to capacity.
- The data rate of user n is
$$R_n = f_s \sum_{k=1}^{K} r_k^n,$$
where f_s is the DMT symbol rate.

The goal of spectrum management problem is to achieve best possible user rates tradeoff among users in the network, i.e., to find the boundary of rate region. Assume that each user is subject to an individual total transmission power constraint. Finding the boundary of rate region is mathematically equivalent to

$$\max_{p_1,p_2,\cdots,p_K} R_1 \tag{5.2}$$
$$s.t. \begin{cases} R_n \geq T_n, \forall n \geq 1; \\ \sum_k p_k^n \leq P_n, \ \forall n, k \\ p_k^n \leq p_k^{n,mask}, \ \forall n, k \end{cases}$$

where

- T_n is minimum target rates of user n.
- P_n is maximum total transmission power of user n.

The spectrum management problem (5.2) aims to maximize the rate of user 1 while guarantees the achievable rates of other users higher than their required minimum target rates T_n. P_n denotes the maximum total transmission power of user n. Spectral mask constraints $p_k^{n,mask}$ may also be applied if needed.

The centralized algorithm based on dual decomposition for OSB, proposed in [Cendrillon et al., 2006], decouples joint optimization across all tones to make the problem solvable per-tone basis. If the rate region is convex, the optimization problem (2) is equivalent to the weighted sum rate optimization:

$$\max_{p_1,\cdots,p_K} \sum_n \omega_n R_n \tag{5.3}$$
$$s.t \begin{cases} \sum_k p_k^n \leq P_n \ \forall n \\ 0 \leq p_k^n \leq p_k^{n,mask} \ \forall k, n \end{cases}$$

where the weight for user 1, ω_1, is set to unity, resulting in the maximization of the rate of user 1; whereas $\omega_n \geq 0$, $n \neq 1$ can be adjusted to guarantee the target rate of user n.

Note that the assumption that the rate region is convex is justified in [Cendrillon et al., 2006] for two-user in DSL system, and the same logic for two-user can be applied to justify the convexity of rate region for multiple-user case. This approximate convexity of rate region is based on the channel correlation between adjacent tones and small tone spacing, and only becomes exact as the tone spacing approaches zero.

5.2.1 DC formulations of SMP (5.3)

First we write Problem (5.3) in the form of a minimization program

$$\begin{aligned} &\min_{p_1,\cdots,p_K} -\sum_n \omega_n R_n \\ &s.t \begin{cases} \sum\limits_k p_k^n \le P_n \ \forall n \\ 0 \le p_k^n \le p_k^{n,mask} \ \forall k,n \end{cases} \end{aligned} \tag{5.4}$$

We have

$$-\sum_n \omega_n R_n = -\sum_n \omega_n \left(f_s \sum_{k=1}^K r_k^n \right) \tag{5.5}$$

For simplicity of notations, let us set

$$p = (p_1, p_2, \cdots, p_K)^T, \ p_k = (p_k^1, p_k^2, \cdots, p_k^N)^T,$$

$$\beta_{k,n} = (\Gamma h_k^{n,1}, \cdots, \Gamma h_k^{n,n-1}, 0, \Gamma h_k^{n,n+1}, \cdots, \Gamma h_k^{n,N})^T,$$

$$\alpha_{k,n} = (\Gamma h_k^{n,1}, \cdots, \Gamma h_k^{n,n-1}, 1, \Gamma h_k^{n,n+1}, \cdots, \Gamma h_k^{n,N})^T,$$

$$h_{k,n}(p_k) = -\omega_n \log_2(\beta_{k,n}^T p_k + \Gamma \sigma_k^n),$$

$$g_{k,n}(p_k) = -\omega_n \log_2(\alpha_{k,n}^T p_k + \Gamma \sigma_k^n).$$

Then the objective function of Problem (5.4) can be expressed as

$$f(p) = f(p_1, p_2, \cdots, p_K) = \sum_{k=1}^K F_k(p_k) \tag{5.6}$$

with

$$F_k(p_k) := \sum_{n=1}^N (g_{k,n}(p_k) - h_{k,n}(p_k)).$$

Since $g_{k,n}$ and $h_{k,n}$ are convex functions, f is a DC function with the following natural DC decomposition:

$$f(p) = g(p) - h(p),$$

where

$$g(p) := \sum_{k=1}^K \sum_{n=1}^N g_{k,n}(p_k), \ h(p) := \sum_{k=1}^K \sum_{n=1}^N h_{k,n}(p_k).$$

However, from numerical point of views, the DCA scheme corresponding to this DC decomposition is not interesting because it requires an iterative algorithm for solving a convex program at each iteration. In an elegant way we introduce a nice DC reformulation of the problem (5.4) for which the resulting DCA is explicitly determined via a very simple formula. Such a DC decomposition of f is inspired by the following result.

Theorem 5.1 *There exists $\rho > 0$ such that the function*

$$h(p) := \frac{1}{2}\rho||p||^2 - f(p) \tag{5.7}$$

is convex on C, the feasible set of (5.4), say
$C := \{p \in \mathbb{R}^{K\times N} \mid \sum_{k=1}^{K} p_k^n \leq P_n,\ 0 \leq p_k^n \leq p_k^{n,mask}\ \ \forall n = 1, 2, \cdots, N; \quad k = 1, 2, \cdots, K\}$.

Proof 5.1 *First, note that the function h is twice differentiable, and its Hessian is $\nabla^2 h = \rho I - \nabla^2 f$. Furthermore we observe that the function h is convex on C if and only if Hessian $\nabla^2 h(p)$ is semi-definite positive on C. The Hessian of f is computed as*

$$\nabla^2 f = diag\left(([\nabla^2 F_k(p_k)]_{(N\times N)})_{k=1,\cdots,K}\right).$$

Thus, we have

$$||\nabla^2 f|| \leq \max_{k=1,\ldots,K} ||\nabla^2 F_k(p_k)||.$$

In other hand, we can calculate $\nabla^2 F_k(p_k)$ as

$$\nabla^2 F_k(p_k) = \sum_{n=1}^{N} (\nabla^2 g_{k,n}(p_k) - \nabla^2 h_{k,n}(p_k)),$$

with

$$\nabla^2 g_{k,n}(p_k) = \omega_n \frac{\alpha_{k,n}(\alpha_{k,n})^T}{\log(2)(\alpha_{k,n}p_k + \Gamma\sigma_k^n)^2}$$

$$\nabla^2 h_{k,n}(p_k) = \omega_n \frac{\beta_{k,n}(\beta_{k,n})^T}{\log(2)(\beta_{k,n}p_k + \Gamma\sigma_k^n)^2}.$$

It is easy to check that $\nabla^2 g_{k,n}(p_k)$ and $\nabla^2 h_{k,n}(p_k)$ are semi-definite positive. Therefore, if

$$\rho \geq \sum_{n=1}^{N} \frac{\omega_n}{\log(2)(\Gamma\sigma_k^n)^2} \max_{i=1,\cdots,N}\{\alpha_{k,n}^i\} \times (\sum_{i=1}^{N} \alpha_{k,n}^i),$$

then

$$\rho \geq \sum_{n=1}^{N} ||\nabla^2 g_{k,n}(p_k)||$$

and so

$$[\rho I - \sum_{n=1}^{N} \nabla^2 g_{k,n}(p_k)] - \ \textit{semi-definite positive.}$$

By the way $\nabla^2 h = \rho I - \nabla^2 f$ is semi-definite positive.

Using the theorem above we get the next DC decomposition of f:

$$g(p) = \frac{1}{2}\rho||p||^2, \ \ h(p) = \frac{1}{2}\rho||p||^2 - f(p),$$

and Problem (5.4) can be now written in the form

$$\min\{\ f(p) := g(p) - h(p) \mid p\ \in C\}$$

or again, in the standard form of DC program:

$$\min\{\chi_C(x) + \ f(p)\ ) \mid p\ \in \mathbf{R}^{N\times K}\}.$$

5.2.2 DCA applied to Problem (5.4)

Starting with $\mathbf{p}^{(0)} = (p_1^{(0)}, \cdots, p_K^{(0)}) \in C$, we have to compute two sequences $\{\mathbf{p}^{(k)}\}$ and $\{\mathbf{q}^{(k)}\}$ such that

$$\mathbf{q}^{(k)} \in \partial h(\mathbf{p}^{(k)}) \text{ and } \mathbf{p}^{(k+1)} \in \partial(g + \chi_C)^*(\mathbf{q}^{(k)}).$$

As indicated above $\mathbf{p}^{(k+1)}$ is an optimal solution of the convex quadratic program

$$\min\{g(p) - \langle p, \mathbf{q}^k \rangle | p \in C\}. \tag{5.8}$$

Then, DCA apply to Problem (5.4) is described as **Algorithm 8**.

Algorithm 8

Initialization:

* Choose an initial point $\mathbf{p}^{(0)} \in C$, set $r := 0$;
* Let ε be a sufficiently small positive number;

Repeat:

* Calculate

$$\mathbf{q}^{(r)} = \nabla h(\mathbf{p}^{(r)}) = \rho\, \mathbf{p}^{(r)} - \nabla f(\mathbf{p}^{(r)})$$

* Calculate $\mathbf{p}^{(r+1)} \in \partial(g + \chi_C)^*(\mathbf{q}^{(r)})$ by solving the linear constrained quadratic program

$$\min\{\frac{1}{2}\rho||p||^2 - \langle p, \mathbf{q}^{(r)} \rangle | p \in C\} \tag{5.9}$$

* Set $r \leftarrow r + 1$;

Until either $||\mathbf{p}^{(r+1)} - \mathbf{p}^{(r)}|| \leq \varepsilon(||\mathbf{p}^{(r)}|| + 1)$ or $|f(\mathbf{p}^{(r+1)}) - f(\mathbf{p}^{(r)})| \leq \varepsilon(|f(\mathbf{p}^{(r)})| + 1)$.

Theorem 5.2 *(Convergence properties of DCA)*

i) **Algorithm 8** *generates a sequence* $\{P^k\}$ *such that the sequence* $\{f(\mathbf{p^k})\}$ *is monotonously decreasing.*

ii) The point $\mathbf{p}^*$ *is a critical point of Problem (5.4).*

5.3 Numerical experiments

In this section, the performance of proposed OSB based on concave minimization is evaluated in various realistic upstream VDSL scenarios with 26-gauge (0.4 mm) lines, tone spacing Δf =4.3125 kHz, DMT symbol rate f_s =4 kHz, and target symbol error probability of 10^{-7} or less. The coding gain and noise margin are set to 3 and 6 dB, respectively. ETSI noise model A [Oksman and Cioffi, 1999] is implemented to model non-VDSL disturbers, consisting of 10 ADSL, 4 HDSL, and 10 ISDN disturbers. In all our simulations, we adopted the FDD band plan 998 [McCammon, 2000], which specifies two separate bands for upstream transmission: 3.75-5.2 MHz and 8.5-12 MHz. The optional 30-138 kHz band is not used. The value of the maximum transmit power equals to 11.5 dBm.

Simulations were test with DCA and SCALE, OSB, IWF, SIW algorithms. The results are presented in Figure 9.2.We can see that even is a local algorithm DCA almost converges to a global solution. The rate region obtains by DCA almost the same with OSB (a global algorithm) and SCALE algorithm.

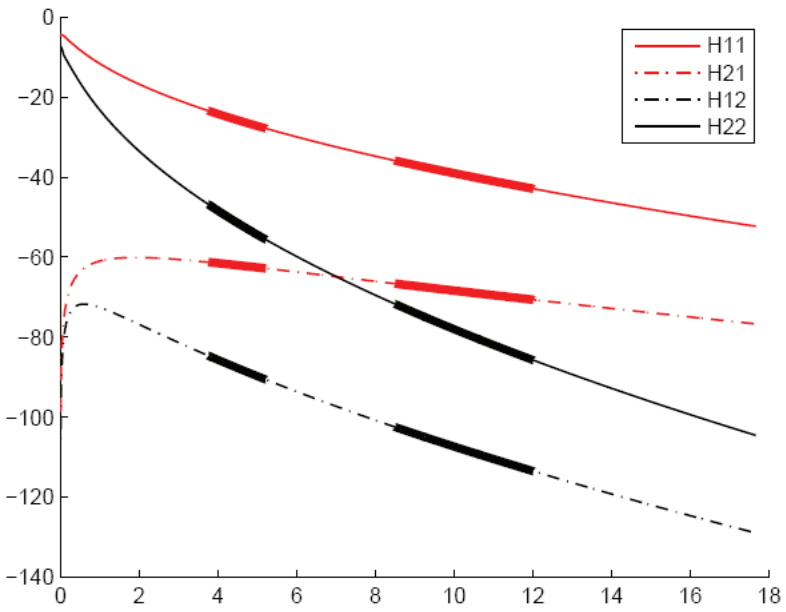

Figure 5.1: Test case for 2 users.

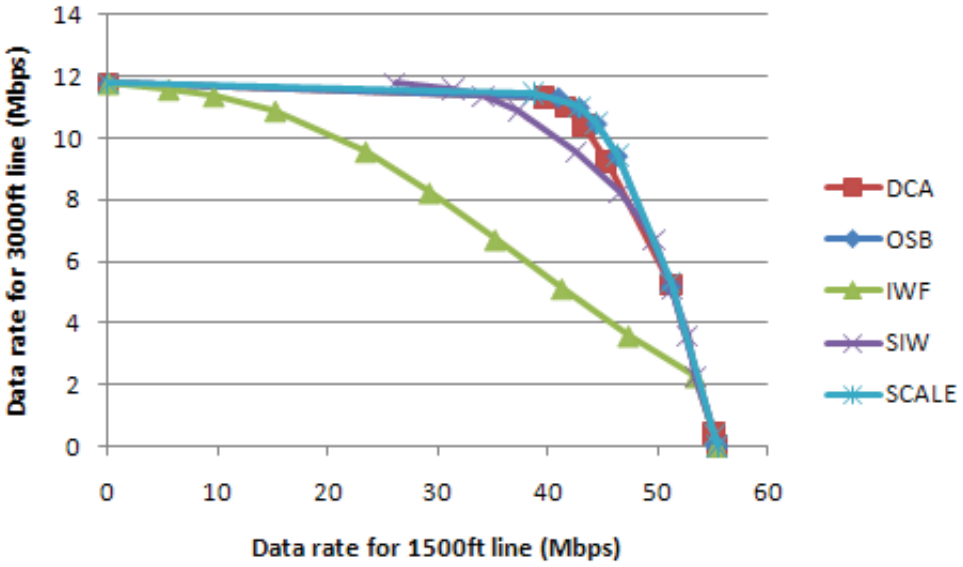

Figure 5.2: Rate region for 2 users, 1148 tones.

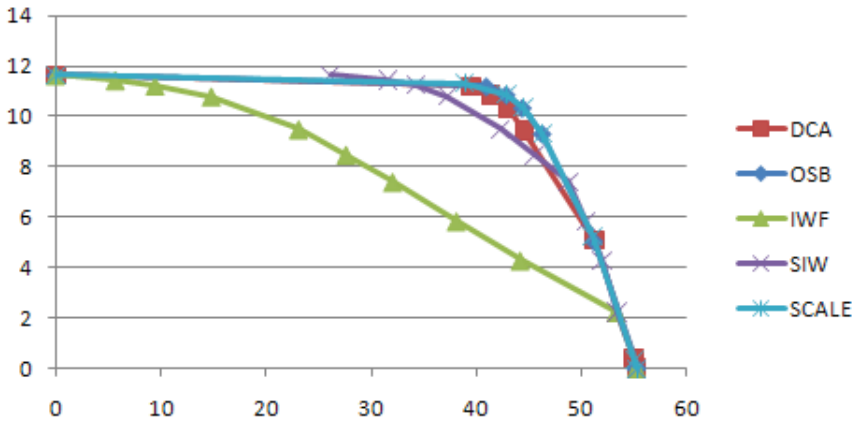

Figure 5.3: Rate region for 2 users, 100 tones.

Other numerical results

We simplify more datasets from the original real data with respect to difference tones. The numerical results are presented in the following to show the efficiency and promising of DCA (see Figure. 9.3,9.4,9.5) in OSB problem.

5.4 Conclusion

In this paper, we presented an algorithm based on DCA and DC programming for solving OSB problem efficiently. The preliminarily numerical results are compared with SCALE which are some standard algorithms for solving SMP. From the result, it is promising to apply DCA or combine DCA with an other algorithms in general case (e.g., the number of user increases and the channel is non-symmetric).

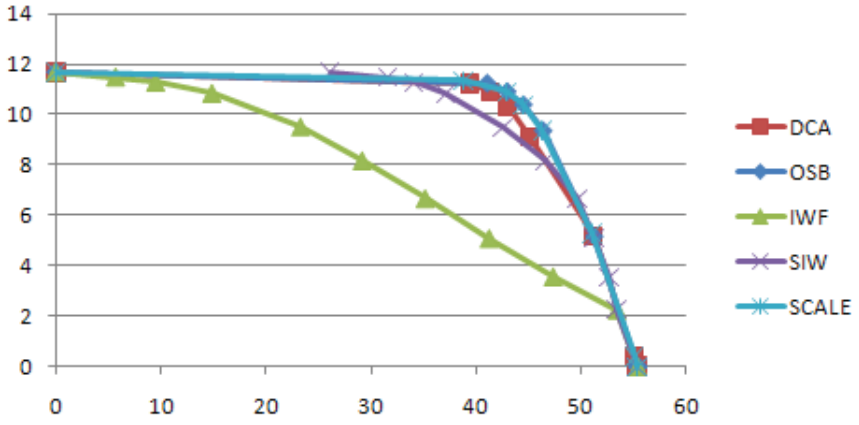

Figure 5.4: Rate region for 2 users, 200 tones.

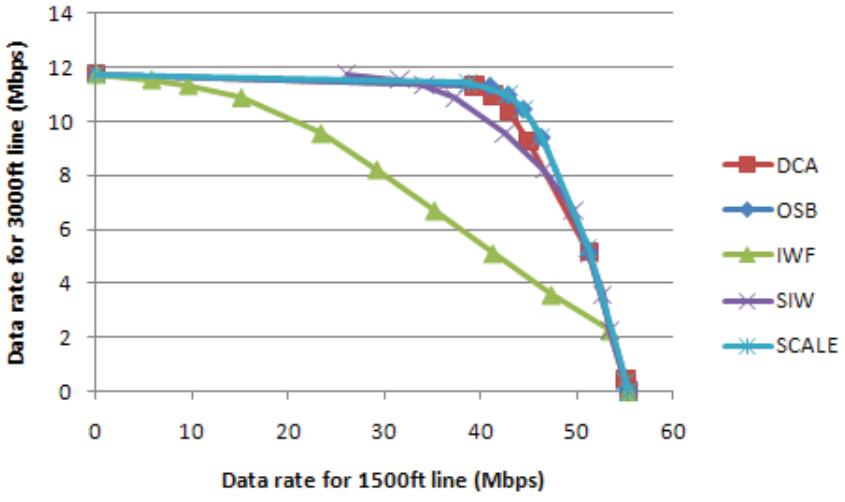

Figure 5.5: Rate region for 2 users, 400 tones.

Part IV

CAR POOLING PROBLEMS

Chapter 6

Solving Car Pooling problem using DCA

This chapter dedicates to introduce and solve Car pooling problem by DCA and Branch and Bound-DCA. The main content of this chapter was taken into account in [Ta et al., 2011].

Car pooling is a well known transport solution that consists in sharing a car between a driver and passengers sharing the same route, or part of it. The challenge is to minimize both the number of required cars and the additional cost in terms of time for the drivers. There are two resulting problems that are interdependent and NP-complete: assigning passengers to cars and finding the shortest path for the drivers so that the overall cost is minimized. In this paper, we present the formulate of Car pooling problem as a Mix Integer Linear Program (MILP) and then investigate a new solution method based on DC (Difference of Convex functions) programming and DCA (DC Algorithms). In order to globally solve the problem, we combine DCA with classical Branch and Bound algorithm (BBDCA). DCA is used to calculate upper bound while lower bound is calculated from a liner relaxation problem. Preliminary numerical results which obtained by DCA and BB-DCA are compared with CPLEX, the best solver for MILP. They show that the proposed algorithm is an efficient algorithm for solving MILP.

6.1 Introduction

Car pooling consists in managing the sharing of a *pool* of cars between several users that have whole or part of their route in common. To solve the problem, several tasks should be performed: choosing drivers and passengers, allocating passengers to cars, computing an optimized route for the cars.

At each step, user preferences or specific constraints may be taken into account, such as the maximum amount of additional time a driver accepts to dedicate to taking passengers, or the time and location constraints for each user such as: departure and arrival place, maximum travel time, etc.

As such, the car pooling transport problem may be described as some kind of fleet management problem, since it consists in optimizing the route of several vehicles taking into account the time constraints of both the drivers and potential passengers. According to a recent survey classifying pickup and delivery problems [Parragh et al., 2008], it is a Vehicle Routing Problem with Pickups and Deliveries (VRPPD), where passengers play the same role as "goods". More precisely, it is a subset of the dial-a-ride problem [Cordeau and Laporte, 2003], where the planning step can be done either a long time prior to the trip – for instance the day before – or in real-time in a demand responsive fashion. In the first case, the quality of the solution would be prioritized, whereas in the second one, speed of execution would be paramount.

There are multiple forms for the car pooling problems, mainly depending on the scenario and use cases considered. If car pooling is managed for the employees of a large company or similarly of group of companies located at the same place, the problem is somehow

simplified since there's only one possible destination with several possible starting points for the passengers and the drivers.

In this paper, we introduce the car pooling problem (CPP) and the formulation of this problem as MILP (see [Baldacci et al., 2004]). As a result in Chapter 1.3, we show how to adapt this approach for solving the car pooling problem in the form of MILP. The proposed DCA for MILP enjoys several advantages: it converges to a local (integer) solution after a finitely many iterations, and requires only the solution of a few number of linear programs. Moreover, it is worth to mention that, although the DCA is a continuous approach working on a continuous domain, it provides an integer solution. This is unusual in continuous approaches and is original property of the proposed DCA.

The numerical results of DCA is compared with BBDCA and CPLEX. The computational results on several test problems show that DCA is an efficient algorithm for solving MILP.

6.2 Problem statement and mathematical model

In this paper, the car pooling problem is considered as the problem which is description in [Baldacci et al., 2004]. The topology of the network is given as a directed graph $G = (V, A)$, where $V = \{0, ..., n\}$ is the set of nodes and A is the set of arcs. The node 0 is associated with the company workplace and the set of nodes $V' = \{1, ..., n\} \subset V$ is corresponding to the employees. Let $V_s = \{1, ..., n_s\}$ (resp. $V_c = \{n_s + 1, ..., n\}$) be the subset of nodes associated with the servers (resp. the clients), so we have $V' = V_s \cup V_c$. The sets of *ongoing* and *incoming* of node $i \in V$ are denoted by Γ_i and Γ_i^{-1}, respectively (i.e., $\Gamma_i = \{j : (i, j) \in A\}$ and $\Gamma_i^{-1} = \{i : (i, j) \in A\}$). Let d_{ij} be a nonnegative cost and t_{ij} be a travel time associated with each arc $(i, j) \in A$.

Each client $i \in V_c$ has an associated penalty p_i representing its contribution to the total cost in case no server picks him up. Each employee $i \in V'$, let e_i be the earliest start time from home and l_i be the latest arrival time to workplace. Associated with each server $k \in V_s$, let Q_k and T_k be the the number of places available on his car and the maximum driving time planing to spend to go from home to workplace.

It is assumed that $t_{k0} \leq T_k \leq l_k - e_k \; \forall k \in V_s$. A simple path $P = (i_1, i_2, \cdots, i_m, i_{m+1})$ in G starting from server $i_1 \in V_s$, visiting clients $\{i_2, ..., i_m\} \subseteq V_c$ and ending at workplace $i_{m+1} = 0$, is called a feasible path if it satisfies the following constraints (see [Baldacci et al., 2004]):

(i) *Capacity constraint:* The number of vertices in P must be not greater than Q_{i_1}.

(ii) *Maximum traveling time constraint:* The total travel time of P must be not greater than T_{i_1}.

(iii) *Departure/arrival time constraint:* Let s_{i_r} be the departure time from vertex i_r in P and let $s_{i_{m+1}}$ be the arrival time at the workplace. Path P satisfies the departure/arrival time constraint if a feasible solution $\{s_{i_1}, s_{i_2}, \cdots, s_{i_{m+1}}\}$ exists to the following inequalities:

$$s_{i_{r+1}} - s_{i_r} \geq t_{i_r i_{r+1}}, \; \forall r = 1, \cdots, m, \tag{6.1}$$

$$s_{i_r} \geq e_{i_r}, \; \forall r = 1, \cdots, m, \tag{6.2}$$

$$l_{i_r} \geq s_{i_{m+1}}, \; \forall r = 1, \cdots, m. \tag{6.3}$$

The CPP is to find n_s feasible paths visiting each employee at most one with respect to minimize the sum of the path costs plus the penalties of unservices clients.

Useful Reduction

To reduce the time of the computational performance, the arcs can be removed from G if that cannot belong to any feasible path. Let sht_{ij} be the value of the shortest time path from i to j. In this paper, the following reductions are considered:

(1) A server cannot pick up another server; hence, remove arcs $(i, j) \in A$ with $i, j \in V_s$.

(2) Server k cannot go directly to pick up client i if the total travel time exceeds the maximum ride time of server k; hence, remove every arc (k, i), with $k \in V_s$ and $i \in V_c$, such that $t_{ki} + sht_{i0} > T_k$.

(3) Server k cannot go directly to pick up client i if he cannot reach the workplace before time $\min[l_k, l_i]$; hence, remove any arc (k, i), with $k \in V_s$ and $i \in V_c$, such that $\max[e_i, e_k + t_{ki}] + sht_{i0} > \min[l_k, l_i]$.

6.2.1 Mathematical formulation

In this section, we present a mathematical formulations of the CPP [Baldacci et al., 2004]. The one is based on three-index decision variables specifying the arcs traversed by each server. We consider four sets of variables:

- for all $(i, j) \in A$ and $k \in V_s$, let us set

$$x_{ij}^k = \begin{cases} 1 \text{ if arc } (i,j) \text{ is traversed by server } k, \\ 0 \text{ otherwise,} \end{cases}$$

- for all $i \in V_c$, let us set

$$y_i = \begin{cases} 1 \text{ if client } i \text{ is not picked up by any server,} \\ 0 \text{ otherwise,} \end{cases}$$

- s_i representing the pickup time of employee $i \in V'$,
- h^k denoting the arrival time of server $k \in V_s$ at the workplace.

A mathematical formulation of the CPP is the following:

$$\min z(F) = \sum_{k \in V_s} \sum_{(i,j) \in A} d_{ij} x_{ij}^k + \sum_{i \in V_c} p_i y_i \tag{6.4}$$

subject to

$$\sum_{j \in \Gamma_k} x_{kj}^k = 1, \ k \in V_s, \tag{6.5}$$

$$\sum_{j \in \Gamma_0^{-1}} x_{j0}^k = 1, \ k \in V_s, \tag{6.6}$$

$$\sum_{j \in \Gamma_i^{-1}} x_{ji}^k - \sum_{j \in \Gamma_i} x_{ij}^k = 0, \ i \in V_c, \ k \in V_s, \tag{6.7}$$

$$\sum_{(i,j) \in A} x_{ij}^k \leq Q_k, \ k \in V_s, \tag{6.8}$$

$$\sum_{(i,j) \in A} t_{ij} x_{ij}^k \leq T_k, \ \ k \in V_s, \tag{6.9}$$

$$s_j - s_i \leq t_{ij} + M(1 - \sum_{k \in V_s} x_{ij}^k), \quad (i,j) \in A, \text{[1]} \tag{6.10}$$

$$s_i \geq e_i, \quad i \in V', \tag{6.11}$$

$$h^k \geq s_i + t_{i0} - M(1 - x_{i0}^k), \quad i \in V', \ k \in V_s, \tag{6.12}$$

$$h^k \leq l_i + M(1 - \sum_{j \in \Gamma_i} x_{ij}^k), \quad i \in V', \ k \in V_s, \tag{6.13}$$

$$\sum_{k \in V_s} \sum_{j \in \Gamma_i} x_{ij}^k + y_i = 1, \quad i \in V_c, \tag{6.14}$$

$$x_{ij}^k \in \{0,1\}, \quad (i,j) \in A, \ k \in V_s, \tag{6.15}$$

$$h^k \geq 0, \ k \in V_s \text{ and } s_i \geq 0, \ i \in V'. \tag{6.16}$$

In this problem, the objective function (6.4) aims to minimize the sum of the costs of the paths used by the servers to reach the workplace plus the penalties cost of the unserviced clients. Equations (6.5) ensure that each server leaves its house, while Equations (6.6) impose that each server arrives at the workplace. Equations (6.7) are continuity constraints. Inequalities (6.8) and (6.9) are capacity and maximum time constraints, respectively. Inequalities (6.10) and (6.11) define the pick up time variables $s_i, i \in V'$, while inequalities (6.12) and (6.13) set the arrival times $h^k, k \in V_s$, of the servers at the workplace and ensure that every employee $i \in V'$ arrives at the workplace at a time compatible with l_i, respectively. Equations (6.14) ensure that each client is either picked up by a server or is left unserviced.

6.3 Solving Car Pooling problem by DCA

6.3.1 DC Algorithm for solving Car Pooling problem

In this section, we consider the car pooling problem (6.4)-(6.16). By using an exact penalty result, we can reformulate the car pooling problem in the form of a concave minimization program. The exact penalty technique aims at transforming the original problem (6.4)-(6.16) into a more tractable equivalent DC program.

Let us set $u = (s,h) \in \mathbb{R}_+^n, v = (x,y) \in \mathbb{R}^m$, matrix $\widetilde{A}$ and vector $\widetilde{b}$ are defined such that the constraints (6.5)-(6.14) can be expressed as $\widetilde{A}(u,v) \leq \widetilde{b}$.

Let

$$K := \{(u,v) \in \mathbb{R}^{n+m} : \widetilde{A}(u,v) \leq \widetilde{b}, \ u \in \mathbb{R}_+^n, \ v \in [0,1]^m\}.$$

The feasible set of (6.4) is then

$$S = \{(u,v) : (u,v) \in K, \ u \in \mathbb{R}_+^n, \ v \in \{0,1\}^m\}.$$

Let us consider the function $p : \mathbb{R}^{m+n} \to \mathbb{R}$ defined by:

$$p(u,v) = \sum_{i=1}^{m} v_i(1 - v_i).$$

It is clear that $p(u,v)$ is concave and finite on K, $p(u,v) \geq 0 \ \forall (u,v) \in K$ and that:

$$\big\{(u,v) : (u,v) \in S\big\} = \big\{(u,v) : (u,v) \in K, p(u,v) \leq 0\big\}.$$

[1] The constraints (9) in [Baldacci et al., 2004] is $s_j - s_i \geq t_{ij} + M(1 - \sum_{k \in V_s} x_{ij}^k), \ (i,j) \in A$, but there is *a text error* in this inequality, in which $\geq$ has to replace by $\leq$ as in (6.10).

Hence problem (6.4) can be rewritten as:

$$\min\Big\{\sum_{i=1}^{m} c_i v_i : (u,v) \in K, p(u,v) \leq 0\Big\}. \tag{6.17}$$

From the penalty results in Theorem 1.6 we get, for a sufficiently large number η ($\eta > \eta_0$), the equivalent concave minimization problem (6.17):

$$\min\{f_\eta(u,v) := \sum_{i=1}^{m} c_i v_i + \eta p(u,v) : (u,v) \in K\},$$

which is a DC program of the form:

$$\min\big\{g(u,v) - h(u,v) : (u,v) \in \mathbb{R}^{n+m}\big\}, \tag{6.18}$$

where

$$g(u,v) = \chi_K(u,v),$$
$$h(u,v) = -f_\eta(u,v) = -\sum_{i=1}^{m} c_i v_i - \eta p(u,v).$$

We have successfully transformed an optimization problem with integer variables into its equivalent form with continuous variables. Notice that (6.18) is a polyhedral DC program where g is a polyhedral convex function (i.e., the pointwise supremum of a finite collection of affine functions).

DCA applied to the DC program (6.18) consists of computing, at each iteration k, the two sequences $\big\{(u^k, v^k)\big\}$ and $\big\{(\alpha^k, \zeta^k)\big\}$ such that $(\alpha^k, \zeta^k) \in \partial h(u^k, v^k)$ and (u^{k+1}, v^{k+1}) solves the next linear program of the form (P_k)

$$\min\{g(u,v) - \langle (u,v) - (u^k, v^k), (\alpha^k, \zeta^k)\rangle : (u,v) \in \mathbb{R}^{n+m}\}$$
$$\Leftrightarrow \min\{-\langle (u,v), (\alpha^k, \zeta^k)\rangle : \ (u,v) \in K\}. \tag{6.19}$$

From the definition of h, a subgradient $(\alpha^k, \zeta^k) \in \partial h(u^k, v^k)$ can be computed as follows:

$$(\alpha^k, \zeta^k) = \nabla h(u^k, v^k) = (0, 2\eta v^k - \eta \mathbf{e} - c), \tag{6.20}$$

where $e = (1, \cdots, 1)^T \in \mathbb{R}^m$ and $c = (c_1, \cdots, c_m)^T$.

The DCA scheme applied to (6.18) can be summarized as **Algorithm 9**.

Algorithm 9 DCA

Initialization:

Choose an initial point (u^0, v^0), set $k = 0$;

Let $\varepsilon_1, \varepsilon_2$ be sufficiently small positive numbers;

Repeat

Compute (α^k, ζ^k) via (6.20);

Solve the linear program (6.19) to obtain (u^{k+1}, v^{k+1});

$k \leftarrow k + 1$;

Until either $\|(u^{k+1}, v^{k+1}) - (u^k, v^k)\| \leq \varepsilon_1(\|(u^k, v^k)\| + 1)$ or $|f_\eta(u^{k+1}, v^{k+1}) - f_\eta(u^k, v^k)| \leq \varepsilon_2(|f_\eta(u^k, v^k)| + 1)$.

Theorem 6.1 *(Convergence properties of Algorithm DCA)*

- *DCA generates the sequence* $\{u^k, v^k\}$ *contained in* $V(K)$ *such that the sequence* $\{f_\eta(u^k, v^k)\}$ *is decreasing.*

- *The sequence* $\{(u^k, v^k)\}$ *converges to* $(u^*, v^*) \in V(K)$ *after a finite number of iterations.*
- *The point* (u^*, v^*) *is a critical point of Problem (6.18). Moreover if* $v_i^* \neq \frac{1}{2}$ *for all* $i \in \{0, \ldots, n\}$*, then* (u^*, v^*) *is a local solution to (6.18).*
- *For a number* η *sufficiently large, if at iteration* r *we have* $v^r \in \{0,1\}^m$*, then* $v^k \in \{0,1\}^m$ *for all* $k \geq r$*.*

Proof 6.1 *See Theorem 1.7.*

6.3.2 A combined DCA-Branch and Bound algorithm

For globally solving the strategic car pooling problem we combine the DCA with the classical Branch and Bound method applied to MILP. The linear relaxation is used for computing lower bounds while the upper bounds are determined by applying DCA to (6.18). The main result is presented in Chapter 1.3.

Our combined algorithm can be summarized as follows: starting with the rectangle $R_0 := [0,1]^m$, we consider at each iteration $k \geq 0$ the rectangle R_k corresponding to the smallest lower bound β_k. The selected rectangle R_k is divided into two subrectangles $R_k, i = 0, 1$ and the lower bound is improved by solving the corresponding linear programs. The upper bound γ_k is determined by applying the DCA to (6.18). The procedure is terminal when $\gamma_k - \beta_k \leq \varepsilon$ and it provides an ε-optimal solution of MILP.

The combined algorithm.

Let $R_0 := [0,1]^m$.
Set $\gamma_0 := +\infty$, $\beta_0 := -\infty$, $restart := true$, $\mathcal{R} := \{R_0\}$, and $k = 0$.
Let ε be sufficiently small positive number.

1. Let R_k be the rectangle such that

$$\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathcal{R}\}.$$

   Bisect R_k into two subrectangles R_{k_0} and R_{k_1} via the index $j*$

$$R_{k_i} = \{v \in R_k : v_{j*} = i, \quad i = 0, 1\}$$

2. Compute lower bounds β_{k_i} $(i = 0, 1)$ by solving the linear relaxation problems corresponding to the set R_{k_i}.

3. If $(restart = true)$ then update γ_k, the best upper bound of the optimal value of (MILP) by applying DCA to Problem (6.18) from a suitable starting point discovered in Step 2.

4. If $\mathcal{R} =$ (i.e. $\gamma_k - \beta_k \leq \varepsilon$), then STOP, the optimal solution is (u^k, v^k) that verify $\langle c, v\rangle = \gamma_k$, otherwise update

$$\mathcal{R} \leftarrow \mathcal{R} \cup \{R_{k_i} : \beta(R_{k_i}) < \gamma_k - \varepsilon, i = 0, 1\} \setminus R_k$$

   and go to Step 1.

The combined algorithm differs from the classical Branch and Bound scheme by Step 3 in which DCA is investigated. Here *restart* is a boolean variable which takes value *true* when we decide to restart DCA.

The question *when DCA is restarted* is interesting from numerical points of view and it will be studied. As in several DC programs, DCA provides a global solution to (6.18)

(and so is to (MILP)) from a *good* starting point. Such a point can be found efficiently while computing lower bounds. We will see in the computational experiments that DCA provides an integer solution after several iterations of the Branch and Bound algorithm. Nevertheless we must continue the Branch and Bound process to ameliorate the lower bound until it is close to the best upper bound. In fact, the Branch and Bound algorithm is introduced to find a good starting point for DCA and check the globality of DCA. *The starting point of DCA* in this case is the solution of linear relaxation problem.

When DCA is restarted?
During the branch and bound process we restart DCA when a feasible solution to (MILP) which improves the best current upper bound is pointed out. In such a case, the starting point of DCA is the just mentioned feasible solution to (MILP).

On the other hand, DCA is also restarted when the number of the 0-1 components of the binary variables (denoted $N_{y^{R_k}}$) of the solution (u^{R_k}, v^{R_k}) to the corresponding linear relaxation problem is sufficiently large, namely $N_{y^{R_k}} \geq m/2$.

We now describe the combined DCA-Branch and Bound algorithm for globally solving Problem (MILP) as **Algorithm 10**.

Algorithm 10 Branch and Bound-DCA (BBDCA)

Initialization:
Let $R_0 := [0,1]^m$.
Solve the linear relaxation problem of (MILP) to obtain an optimal solution (u^{R_0}, v^{R_0}) and the first lower bound $\beta_0 := \beta(R_0)$.
Solve (6.18) by DCA from the starting point (u^{R_0}, v^{R_0}) to obtain $(u_t^{R_0}, v_t^{R_0})$.
If $(u_t^{R_0}, v_t^{R_0})$ is feasible to (MILP) **then** set $\gamma_0 := \langle c, v_t^{R_0} \rangle$ and set $(u^0, v^0) := (u_t^{R_0}, v_t^{R_0})$
else set $\gamma_0 := +\infty$.
If $(\gamma_0 - \beta_0) \leq \varepsilon|\gamma_0|$ **then** (u^0, v^0) is an ε-optimal solution of (MILP)
else set $\mathcal{R} \leftarrow \{R_0\}, k \leftarrow 0$.
While stop = false **do**
Select a rectangle R_k such that $\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathcal{R}\}$.
Bisect R_k into two subrectangles R_{k_0} and R_{k_1} via the index $j*$
$R_{k_i} = \{v \in R_k : v_{j*} = i, \quad i = 0, 1\}$
Solve the subproblems (P_{k_i}) to obtain $\beta(R_{k_i})$ and $(u^{R_{k_i}}, v^{R_{k_i}})$, (i = 0, 1):
$(P_{k_i}) \quad \beta(R_{k_i}) := \min\{\langle c, v\rangle : (u, v) \in K,\ v \in R_{k_i}\}$.
If $(u^{R_{k_i}}, v^{R_{k_i}})$ is the best feasible solution to (MILP) **then**
update γ_k and the best feasible solution (u^k, v^k) by applying DCA to (6.18) from $(u^{R_{k_i}}, v^{R_{k_i}})$.
else if $(N_{y^{R_{k_i}}} \geq m/2)$ **then**
Solve the linear relax problem to obtain $(u_t^{R_{k_i}}, v_t^{R_{k_i}})$.
Apply DCA to (6.18) from $(u_t^{R_{k_i}}, v_t^{R_{k_i}})$.
Update γ_k and the best feasible point (u^k, v^k).
Endif
Endif
Set $\mathcal{R} \leftarrow \mathcal{R} \cup \{R_{k_i} : \beta(R_{k_i}) < \gamma_k - \varepsilon, i = 0, 1\} \setminus R_k$
If $\mathcal{R} = \emptyset$ **then** STOP, (u^k, v^k) is ε-optimal solution, **else** $k \leftarrow k + 1$.
Endwhile

	DCA			BBDCA			CPLEX12.2			
DATA	OBJ	Time	GAP	OBJ	Time	GAP	OBJ	Time	GAP1	GAP2
DATA1	4340	0,17	6,93	4340	0,28	6,93	4190	0,13	3,46	3,46
DATA2	3908	0,19	2,66	3908	0,27	2,66	3869	0,15	1.00	1.00
DATA3	4529	0,17	9,29	4529	0,28	9,29	4301	0,11	5,03	5,03
DATA4	4411	0,19	3,42	4411	0,27	3,42	4382	0,18	0,66	0,66
DATA5	4414	0,17	9,64	4243	18,47	6,00	4188	0,19	5,12	1,30
DATA6	3883	0,19	5,44	3827	5,47	4,06	3815	0,17	1,75	0,31
DATA7	4335	0,19	5,45	4308	177,82	4,86	4244	0,19	2,10	1,49
DATA8	4046	0,17	6,70	3907	0,73	3,38	3888	0,18	3,91	0,49
DATA9	4436	0,27	9,37	4436	0,36	9,37	4250	0,19	4,19	4,19
DATA10	4259	0,18	4,24	4259	0,27	4,24	4215	0,18	1,03	1,03
DATA11	3773	0,17	6,82	3773	0,28	6,82	3639	0,19	3,55	3,55
DATA12	4384	0,17	6,94	4384	0,25	6,94	4246	0,17	3,15	3,15
DATA13	4035	0,19	5,74	4035	0,28	5,74	3882	0,17	3,79	3,79
DATA14	4123	0,19	5,31	4028	0,64	3,08	3982	0,18	3,42	1,14
DATA15	4293	0,17	5,88	4293	0,27	5,88	4201	0,18	2,14	2,14
DATA16	4299	0,17	4,94	4299	0,27	4,94	4263	0,19	0,84	0,84
DATA17	4474	0,16	4,73	4474	0,26	4,73	4390	0,15	1,88	1,88
DATA18	4112	0,19	7,41	4060	7,75	6,22	4018	0,15	2,29	0,99
DATA19	4018	0,28	3,37	4018	0,38	3,37	3998	0,31	0,50	0,50
DATA20	4112	0,19	5,88	4085	0,64	5,26	4059	0,21	1,30	0,64
Average	4209,2	0,19	6,00	4180,8	10,76	5,36	4101	0,18	2,55	1,88

Table 6.1: Comparative results between DCA, BBDCA and Cplex m=51, n ≈ 400.

	DCA			BBDCA			CPLEX12.2			
DATA	OBJ	Time	GAP	OBJ	Time	GAP	OBJ	Time	GAP1	GAP2
DATA21	9010	0,64	7,60	9010	0,97	7,6	8516	0,45	5,48	5,48
DATA22	8075	0,72	8,23	8075	1,05	8,23	7665	0,91	5,08	5,08
DATA23	9620	0,64	7,44	9238	154,92	3,61	9142	0,47	4,97	1,04
DATA24	7990	0,63	6,59	7804	37,17	4,36	7718	0,65	3,40	1,10
DATA25	9333	0,66	6,1	9333	0,95	6,1	9012	0,61	3,44	3,44
DATA26	8572	0,64	8,25	8572	1,00	8,25	8084	0,71	5,69	5,69
DATA27	8331	0,64	5,81	8331	1,02	5,81	8016	0,52	3,78	3,78
DATA28	9167	0,64	8,4	9167	0,97	8,4	8754	0,61	4,51	4,51
DATA29	8873	0,66	6,34	8623	2,34	3,63	8528	0,53	3,89	1,10
DATA30	8589	0,67	8,17	8194	67,05	3,75	8168	0,82	4,90	0,32
DATA31	8280	0,64	6,75	8280	0,95	6,75	7908	0,51	4,49	4,49
DATA32	8210	0,66	7,64	8210	1,00	7,64	7835	0,59	4,57	4,57
DATA38	8994	0,75	7,06	8759	22,89	4,57	8652	0,76	3,80	1,22
DATA39	8782	0,72	8,06	8782	1,08	8,06	8439	0,45	3,91	3,91
DATA40	8752	0,72	5,38	8752	1,14	5,38	8583	0,55	1,93	1,93
DATA41	8268	0,78	6,24	8268	1,15	6,24	8029	0,68	2,89	2,89
DATA42	8738	0,78	7,28	8738	1,14	7,28	8497	0,52	2,76	1,22
DATA43	8146	0,75	6,09	7915	2,58	3,35	7849	0,79	3,65	0,83
DATA44	8274	0,73	7,06	8034	101,18	4,29	7912	0,75	4,38	1,52
DATA45	8992	0,76	6,49	8992	1,15	6,49	8674	0,51	3,54	3,54
DATA46	8980	0,72	7,45	8980	1,15	7,45	8695	0,55	3,17	1,33
DATA47	9102	0,75	6,06	9102	1,41	6,06	8772	0,72	3,63	3,63
DATA48	9268	0,76	7,39	8968	17,68	4,30	8812	0,49	4,92	1,74
DATA49	8940	0,78	7,87	8940	1,16	7,87	8578	0,88	4,05	4,05
DATA50	9584	0,76	6,89	9392	112,44	5,02	9324	0,59	2,71	0,32
Average	8754,8	0,70	7,07	8658,36	21,41	6,01	8406,5	0,63	3,98	2,88

Table 6.2: Comparative results between DCA, BBDCA and Cplex m=101, n ≈ 800.

6.4 Numerical experiment

In this section, the results furnished by DCA and Brand and Bound-DCA (BBDCA) are compared with CPLEX 12.2. The algorithm has been coded in C++ and implemented on a Intel Core 2 CPU 2.53 Ghz, RAM 2GB. The directed graph $G = (V, E)$ is randomly generated with m-nodes and n-arcs. The datasets are randomly generated in a similar way as the data used in [Baldacci et al., 2004]. For Brand and Bound algorithm, the STOP condition is setup by maximize number of iteration equaled 1000 or the GAP is less than 10%.

	DCA			BBDCA			CPLEX12.2			
DATA	OBJ	Time	GAP	OBJ	Time	GAP	OBJ	Time	GAP1	GAP2
DATA51	10277	1,14	7,53	10277	1,83	5,48	9951	1,29	3,17	1,02
DATA52	10208	1,10	7,95	10208	1,59	7,95	9761	0,86	4,38	0,83
DATA53	11283	1,13	6,59	11283	1,52	6,59	10932	0,79	3,11	3,11
DATA54	10718	1,03	7,01	10718	1,55	7,01	10312	1,31	3,79	3,79
DATA55	10895	1,05	7,33	10895	1,58	7,33	10472	2,05	3,88	3,88
DATA56	10886	1,02	5,91	10526	96,31	2,69	10428	0,75	4,21	0,93
DATA57	11021	0,98	7,92	11021	1,56	7,92	10496	0,84	4,76	4,76
DATA58	11201	1,03	4,86	10989	96,64	3,04	10962	0,68	2,13	0,25
DATA59	10689	1,02	6,91	10689	1,50	6,91	10285	0,89	3,78	3,78
DATA60	10438	1,05	8,05	10438	1,58	8,05	9915	1,09	5,01	5,01
DATA61	11046	1,03	7,13	11046	1,52	7,13	10537	1,01	4,61	4,61
DATA62	11463	1,05	9,25	11463	1,55	9,25	10881	0,85	5,08	5,08
DATA63	11239	1,00	7,03	11239	1,58	7,03	10718	0,81	4,64	4,64
DATA64	10591	1,05	4,76	10591	1,56	4,76	10296	1,03	2,79	2,79
DATA65	11079	1,02	6,76	10728	101,79	3,71	10645	0,84	3,92	0,77
Average	10868,9	1,05	7,00	10807,4	20,91	6,46	10439,4	1,00	3,95	3,40

Table 6.3: Comparative results between DCA, BBDCA and Cplex m=126, n $\approx$ 1000.

	DCA			BBDCA			CPLEX12.2			
DATA	OBJ	Time	GAP	OBJ	Time	GAP	OBJ	Time	GAP1	GAP2
DATA33	13328	1,42	6,92	13328	2,14	6,92	12672	1,11	4,92	4,92
DATA34	12374	1,41	8,14	12374	2,13	8,14	11819	1,38	4,49	4,49
DATA35	13056	2,13	6,46	12590	184,50	3,00	12497	1,33	4,28	0,74
DATA36	12297	1,45	5,44	12297	2,19	5,44	11948	1,32	2,84	2,84
DATA37	12162	1,48	9,63	12162	2,20	9,63	11406	1,58	6,23	6,23
DATA66	12579	1,50	6,10	12270	164,29	3,74	12181	2,14	3,16	0,73
DATA67	13235	1,13	7,02	13235	1,73	7,02	12785	1,72	3,40	3,40
DATA68	13138	1,47	5,45	12818	341,41	3,09	12732	0,99	3,09	0,67
DATA69	12907	1,47	9,27	12907	2,22	9,27	12178	1,75	5,65	5,64
DATA70	12735	1,86	9,35	12735	2,50	9,35	12123	1,78	4,81	4,81
DATA71	12056	1,17	5,73	11924	98,21	4,71	11823	1,01	1,93	0,62
Average	12715,2	1,50	7,23	12603,6	73,04	6,37	12196,7	1,47	4,07	3,19

Table 6.4: Comparative results between DCA, BBDCA and Cplex m=151, n $\approx$ 1200.

The results obtained by DCA, BBDCA and CPLEX 12.2 are shown in the same tables. We use the following notations:

- $\diamondsuit$ DATA: name of generated problem (For example, for the "DATA1", it consists of the generated data for a problem, where $m = 51, n = 400, n_s = [(m-1)/4]+2$ and the values of another parameters are given in a file "data1.txt".),
- $\diamondsuit$ OBJ: the objective function of each problem,
- $\diamondsuit$ Time: CPU time in second of each algorithm,
- $\diamondsuit$ $GAP = \frac{OBJ-LB}{OBJ}.100\%$ where LB is the best lower bound and OBJ is the objtive function of each problem,
- $\diamondsuit$ $GAP1 = \frac{OBJ_{\text{obtained by DCA}}-OBJ_{\text{obtained by CPLEX}}}{OBJ_{\text{obtained by DCA}}}.100\%$,
- $\diamondsuit$ $GAP2 = \frac{OBJ_{\text{obtained by BBDCA}}-OBJ_{\text{obtained by CPLEX}}}{OBJ_{\text{obtained by BBDCA}}}.100\%$.

From the numerical results, we observe that:

- DCA always provides an integer solution and it converges after a few number of iterations.
- In BBDCA, the DCA with restarting provides a feasible solution to (MILP) very rapidly.

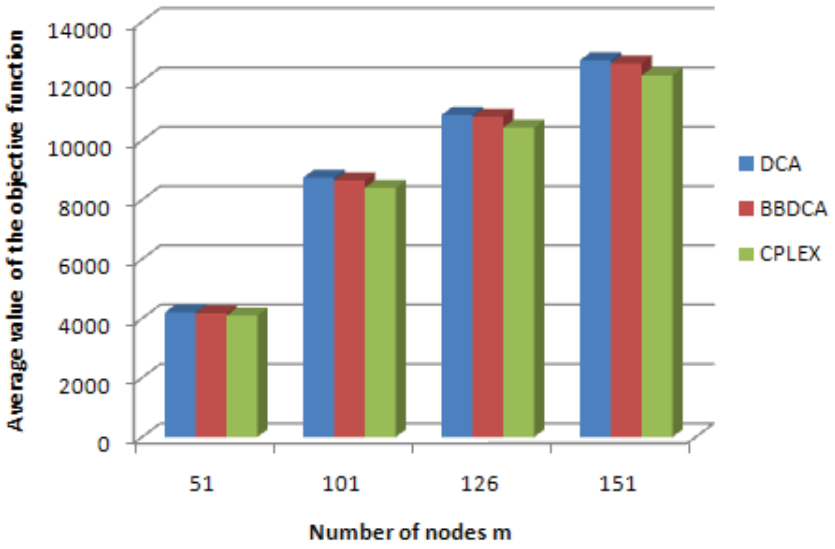

Figure 6.1: Comparison the average value of objective functions between DCA, BBDCA and CPLEX 12.2 with m=51, 101, 126, 151.

- The results of BBDCA is slight better than of DCA but the CPU time of DCA is better than of BBDCA.
- CPLEX12.2 is also slight better than DCA and BBDCA including CPU time and the value of the objective function. However, we note that CPLEX12.2 (the latest version of CPLEX) is combined by many efficient algorithms to solve MILP while DCA is only a local algorithm.

6.5 Conclusion

In this paper, we present MILP formulation for Car pooling problems. An efficient approach based on DC programming and DCA is proposed for solving this problem. The combine DCA-Branch and Bound Algorithm (BBDCA) is given to globally solve problem. Preliminary numerical simulations show that the BBDCA is very interesting. The DCA is original because it can give a mixed integer solution while it works in a continuous domain. Its inexpensiveness is crucial to high-dimensional problems that we address in future work.

Chapter 7

Solving Multiobjective Dynamic Car Pooling problem

This chapter devotes to introduce a particular Car pooling problem in the form of Multiobjective optimization problem. It is an extension version of Car pooling problem in Chapter 6. Then, we proposed a heuristic algorithm to solve the multiobjective dynamic car pooling problem. The main content of this chapter is presented in [Ta et al., 2012b].

Car pooling problem (CPP) is a well known transport solution that consists in sharing a car between a driver and passengers sharing the same route, or part of it. The challenge is to minimize both the number of required cars and the additional cost in terms of time for the drivers. There are two resulting problems that are interdependent and NP-complete: assigning passengers to cars and finding the shortest path for the drivers so that the overall cost is minimized. In this paper, we consider a multiobjective dynamic car pooling problem, where both the cost and the total travel time of drivers are to be minimized. We based on labeling algorithms for solving the multiobjective shortest problem investigate a new algorithm to solve this problem. Preliminary numerical results in a real scenario are reported to show the efficiency of our proposed algorithm and promising to apply in real time applications.

7.1 Introduction

Car pooling consists in managing the sharing of a *pool* of cars between several users that have whole or part of their route in common. To solve the problem, several tasks should be performed: choosing drivers and passengers, allocating passengers to cars, computing an optimal route for the cars.

At each step, user preferences or specific constraints can be considered, such as the maximum travel time a driver accepts to dedicate to taking passengers, or the time and location constraints of each user such as: earliest departure time and latest arrival time at workplace, etc.

To specify, when the role of passenger is considered as "goods", the car pooling problem may be described as some kinds of fleet management problem. According to a recent survey classifying pickup and delivery problems [Parragh et al., 2008], it is a Vehicle Routing Problem with Pickups and Deliveries (VRPPD). More precisely, it is a subset of the dial-a-ride problem [Cordeau and Laporte, 2003], [Cordeau and Laporte, 2007], [Ropke et al., 2007], vehicles routes and schedules are designed for users who specify pickup and delivery requests between origins and destinations where the planning step can be done either a long time prior to the trip – for instance the day before – or in real-time in a demand responsive fashion. In the first case, the quality of the solution would be prioritized, whereas in the second one, the speed of execution would be paramount. However, the major difference between the dial-a-ride problems and the CPP is the vehicle ownership. In the former dial-a-ride problems, there must be full time drivers that service all of the passengers. In the CPP, the vehicles belong to the participants, who take turns sharing use of their cars

and act as drivers to pick up the others. In other words, the drivers are also subject to a transportation demand.

There are multiple forms for the car pooling problems, mainly depending on the scenario and use cases considered. It usually focused on the to-work problem (from different origins to a common destination) or the return-from-work problem (from the same origin to different destinations).

In [Baldacci et al., 2004], [Ta et al., 2011], the car pooling problem is managed for the employees of a large company or similarly of group of companies located at the same place, the problem is somehow simplified since there's only one possible destination with several possible starting points for the passengers and the drivers. In [Baldacci et al., 2004], an exact and a heuristic method are proposed, based on two integer programming formulations, for solving the problem. Their exact method is based on a bounding procedure that combines three lower bounds derived from different relaxations of the problem. A valid upper bound is obtained by the heuristic method, which transforms the Lagrangian lower bound solution into a feasible solution. In [Ta et al., 2011], based on DC programming, DC Algorithm and using penalty technique to solve Binary Integer Linear Program, the two algorithms are presented (DCA) for solving the problem and DCA-Branch and Bound to globalize solution of DCA. In [Calvo et al., 2004], the authors presented an integrated system for the organization of a car pooling service, using several current Information and Communication Technologies (ICT) such as: web, GIS and SMS. The core of the system is an optimization module which solves heuristically the specific routing problem.

However, to the best of our knowledge, there has not yet been any study of car pooling as a multiobjective optimization problem. In this paper, we consider CPP as a bi-objective optimization problem. The problem consists in finding a solution for drivers and passengers sharing their cars with respect to bi-objective: minimize cost (travel cost + penalty cost of unservices clients) and minimize the total travel time of the drivers. Based on labeling algorithm for solving multiobjective shortest path problem, we investigate a new algorithm to solve this problem. The preliminary numerical results of a real scenario are presented to show the efficiency of the proposed algorithm.

The paper is organized as follows. In Section 2 we introduce the multiobjcetive car pooling problem (MCPP). The algorithm is developed in Section 3 while the numerical results are reported in Section 4. Finally, the conclusions are presented in Section 5.

7.2 Problem statement

Herein, the car pooling problem is considered as a generalization of the CPP problem which is description in [Baldacci et al., 2004] and in [Ta et al., 2011]. Two objective functions are considered in MCPP are minimize the cost (travel cost + penalty cost of unservices clients) and minimize total travel time.

The topology of the network is given as a directed graph $G = (V, A)$, where $V = \{0, 1, ..., n\}$ is the set of nodes and A is the set of arcs. The node 0 is associated with the company workplace and the set of nodes $V' = \{1, ..., n_\ell\} \subset V$ $(n_\ell \leq n)$ is corresponding to the employees (drivers and passengers). Let $V_s = \{1, ..., n_s\}$ (resp. $V_c = \{n_s + 1, ..., n_\ell\}$) be the subset of nodes associated with the drivers (resp. the passengers), so we have $V' = V_s \cup V_c \subseteq V$.

Each passenger $i \in V_c$ has an associated penalty p_i representing its contribution to the total cost in case no driver picks him up. Each employee $i \in V'$, let e_i be the earliest start time from home and l_i be the latest arrival time to his destination. Associated with each driver $k \in V_s$, let Q_k and T_k be the the number of available places on his car and the maximum driving time planing to spend to go from home to his destination.

It is assumed that $t_{k0} \leq T_k \leq l_k - e_k \; \forall k \in V_s$. A simple path $P = (i_1, i_2, \cdots, i_m, i_{m+1})$

in G starting from server $i_1 \in V_s$, visiting vertices $\{i_2, ..., i_m\} \subseteq V \backslash \{0\}$ and ending at destination $i_{m+1} = 0$, is called a feasible path if it satisfies the following constraints:

(i) *Capacity constraint:* The number of passengers in P must be not greater than Q_{i_1}.

(ii) *Maximum traveling time constraint:* The total travel time of P must be not greater than T_{i_1}.

(iii) *Departure/arrival time constraint:* Let s_{i_r} be the departure time from vertex i_r (associated with the employee at i_r) in P and let $s_{i_{m+1}}$ be the arrival time at his destination. Path P satisfies the departure/arrival time constraint if a feasible solution $\{s_{i_1}, s_{i_2}, \cdots, s_{i_{m+1}}\}$ exists to the following inequalities:

$$s_{i_{r+1}} - s_{i_r} \geq t_{i_r i_{r+1}}, \ \forall r = 1, \cdots, m, \tag{7.1}$$

$$s_{i_r} \geq e_{i_r}, \ \forall r = 1, \cdots, m, \text{ and } i_r \in V' \tag{7.2}$$

$$l_{i_r} \geq s_{i_{m+1}}, \ \forall r = 1, \cdots, m, \text{ and } i_r \in V'. \tag{7.3}$$

The MCPP is to find n_s feasible paths with respect to bi-objective: minimize the sum of the path costs plus the penalties of unservices passengers and minimize the total travel time of the drivers.

Moreover, the system can be changed by incoming (or leaving) of new drives (cars) or passengers. Thus, for solving this problem, it is necessary to investigate an inexpensive and distributed algorithm. From the practical point of view, we propose a new algorithm, in spite of consider all the drivers at the same time, we consider one driver each time, then try to find *an efficient path (a Pareto point)* corresponding with the driver which minimizes the path costs plus the penalties of unservices passengers and minimizes the total travel time of the driver.

7.3 Algorithm description

The inspiration of our algorithm is based on labeling algorithms (e.g., Martin algorithm,...) for solving multiobjective shortest path problems (see in [Ehrgott, 2005]). The idea of our algorithm is quite simple. Firstly, we take the driver whose departure time is smallest (earliest). Secondly, from this driver, we find a feasible path with respect to minimize the cost (path cost + penalty cost of unservices clients) and minimize the total travel time, it is called *an efficient path*. Thirdly, we remove the driver and its passengers into *the solution list* and restart with a new driver (who have new smallest departure time) and passengers (the rest of passengers). At each restart time, we update new drivers and passengers.

For finding an efficient path, at each iteration, at each vertex, we consider two different sets of labels: *permanent labels* ($\mathcal{PT}$) and *temporary labels* ($\mathcal{TL}$). The algorithm selects the minimum *dominant lexicographic* label from all the sets of temporary labels, converts it to a permanent label, and propagates the information contained in this label to all the temporary labels of its successors. The procedure stops when there are no more temporary labels. A label L_i can be defined as follows:

$$L_i = (c_i, \ t_i, \ n_i, \ T_i, \ j, \ h, \ ID_i) \tag{7.4}$$

where

- c_i is the cost,
- t_i is the departure time,
- n_i is the available places,

- T_i is the latest arrival time,
- j is the adjacent vertex from which it was possible to label the vertex i,
- h is the identification of the label in the list of labels on vertex j from which it created the label L_i,
- ID_i is the identification of label L_i.

At each driver, *the departure time, the maximal travel time, the number of available places* are known, supposed that the latest arrival time of the driver equals to the departure time plus maximal travel time of the driver. At each passenger, *the penalty cost, the earliest departure time, the latest arrival time* are known and each arc associates with a vector (*arc cost, arc travel time*). At each vertex, we define a *dominant relation* between two labels $L_i = (c_i, t_i, n_i, T_i, -, -, -)$ and $L_j = (c_j, t_j, n_j, T_j, -, -, -)$ as follows: we said that L_j is dominated by L_i if and only if $c_i \leq c_j$, $t_i = t_j$, $n_i \geq n_j$, $T_i \geq T_j$ and denote $L_i \lhd L_j$. At one vertex, a label L_i is called *dominance label* if and only if there don't exist a label L_j such that L_i is dominated by L_j.

The algorithm can be presented as Algorithm 11.

Algorithm 11

Initialization:

Read input data: a digraph $G = (V, A)$, with *(travel cost, travel time)* arc vectors; number of drivers; number of passengers; departure time and maximal travel time and number of available places of drivers; penalty cost, earliest departure time and latest arrival time of passengers.

Repeat

Finding a server s with the minimal departure time;

Calculate label L at server s and let $\mathcal{TL} := \{L\}$;

While $\mathcal{TL} \neq \emptyset$ **do**

Let label $L = (c, t, n, T, v_q, h, k)$ of vertex v_i be the lexicographically smallest label in $\mathcal{TL}$;

Remove L from $\mathcal{TL}$ and add it to $\mathcal{PL}$;

For all $v_j \in V$ such that $(v_i, v_j) \in A$ do

Create set of labels (see Create label procedure) $L' = (c', t', n', T', v_i, k, r)$ as the next label at node v_j and add it to $\mathcal{TL}$;

Sort label at v_j by lexicographic order;

Delete all labels of node v_j dominated by L', delete L' if it is dominated by another label of node v_j

End for;

End while

At the destination node

Delete label $(c_j, t_j, -, -, -)$ if there exists a label $(c_i, t_i, -, -, -)$ such that $c_j \geq c_i$ and $t_j \geq t_i$;

Use the predecessor labels in permanent labels to recover all efficient paths from s to destination node, choose an efficient path as the solution (depend on user's criteria);

Remove driver s (resp. its passengers) from set of drivers (resp. from set of passengers) into the solution list;

Until (Number of servers (drivers) = 0)

The algorithm is illustrated by the following example. Let us consider the network in Figure 7.1 with the goal of computing an efficient path from driver 1s and driver 5s to destination 6d, and consider three passengers 2c,3c,4c. For more details, driver 1s (resp.

Algorithm 12 Create labels procedure

Initialization:

$(v_i, v_j) \in A$ and start from label $L = (c, t, n, T, v_q, h, k)$ of vertex v_i (see in Algorithm 11)

If $(T \geq t + t_{ij}))$

Create label $L_1' = (c + c_{ij}, t + t_{ij}, n, T, v_i, k, r_1)$ and add it to $\mathcal{TL}$;

If (v_j **has a passenger and** $n \geq 1$)

If $(e_i \leq t + t_{ij} \leq l_i)$

Create label $L_2' = (c + c_{ij} - p_j, t + t_{ij}, n - 1, \min\{T, T_j\}, v_i, k, r_2)$ and add it to $\mathcal{TL}$;

5s) depart at time 10 (resp. 12), maximal travel time is 9 (resp. 10), number of available places is 2 (resp. 2). Firstly, we have to find the minimum of departure time of driver 1s and driver 5s, it equals to 10 associated with driver 1s. Thus, we start computing at driver 1s with label (27, 10, 2, 19,-, -,1), the cost value equals to the sum of penalty costs of all clients (passengers). From driver 1s, we can visit passengers 2c and 3c.

At passenger 2c, we have *(penalty cost, earliest departure time, latest arrival time) = (6, 12, 17)*. To visit passenger 2c, we have one label at vertex 2c: (32, 13, 2, 19,1,1,1), the cost equals to the cost in previous label plus cost path (1s-2c), it is 27+5= 32, the departure time at 2c is 10 + 3 = 13 (depart time at 1s plus travel time from 1s to 2c), the number of available places is 2 (in this case, the driver goes to visit passenger 2c but he does not take passenger 2c), then the latest arrival time is 19 (it equals to latest arrival time in previous label).

To continue, we test the conditions *departure time at 1s plus travel time from 1s to 2c is belong to the interval: [earliest departure time, latest arrival time] of passenger 2c and the number of place available is greater or equal 1.* If the answer is TRUE, then the driver can take the passenger, otherwise the driver cannot. The inequality in this case is $12 \leq 10 + 3 \leq 17$, the number of available place is 2, then the answer is TRUE. Thus, driver 1s can take the passenger 2c. Thus, a new label is (26, 13, 1, 17,1,1,2), the cost equals to the cost in previous label (1s) plus cost path (1s-2c) minus penalty cost (2c), the latest arrival time equals to the minimum of latest arrival time of driver (1s) (at previous label) and the latest arrival time of passenger (2c), in this case, the latest arrival time equals to $\min\{19, 17\} = 17$, the number of available places is also updated. After that, we sort the label at node (2c) by lexicographic order $(c_i, t_i, n_i, T_i, -, -, -)$.

Similarity, when the driver (1s) visits passenger (3c), we have two labels (31,12,2,19,1,1,1) and (21,12,1,18,1,1,2). Next step, when the driver at (2c), he can visit vertex (3c) and (5). Firstly, he visits vertex (3c) then the labels at vertex (3c) are (27,14,1,17,2,2,3), (17,14,0,17,2,2,4) and (33,14,2,19,2,1,5), (23,14,1,18,2,1,6). Label (27,14,1,17,2,2,3) is dominated by label (23,14,1,18,2,1,6) so we delete the label (27,14,1,17,2,2,3) from the label list at vertex (3c). Now, in vertex (3c), there are five dominance labels and we sort these labels as (17, 14, 0, 17, 2, 2, 4), (21, 12, 1, 18, 1, 1, 2), (23, 14, 1, 18, 2, 1, 6), (31, 12, 2, 19, 1, 1, 1) and (33, 14, 2, 19, 2, 1, 5). When the driver at (2c) visits vertex (5), two labels are updated (30,14,1,17,2,2,1) and (36,14,2,19,2,1,2).

From vertex 5, we can visit vertex (3c), then four labels are created: (32, 17, 1, 17, 5, 1, 7), (22, 17, 0, 17, 5, 1, 8), (38, 17, 2, 19, 5, 2, 9) and (28, 17, 1, 18, 5, 2, 10). Label (32, 17, 1, 17, 5, 1, 7) is dominated by label (28, 17, 1, 18, 5, 2, 10) so we delete it from the label list.

From (3c), the driver can visit passenger (4c). For visiting passenger (4c), the labels are expressed as (20, 16, 0, 17, 3, 4, 1), (24, 14, 1, 18, 3, 2, 2), (26, 16, 1, 18, 3, 6, 3), (34, 14, 2, 19, 3, 1, 4) and (36, 16, 2, 19, 3, 5, 5).

When the driver at vertex (4c), it can go to the destination (6d), so the labels at (6d)

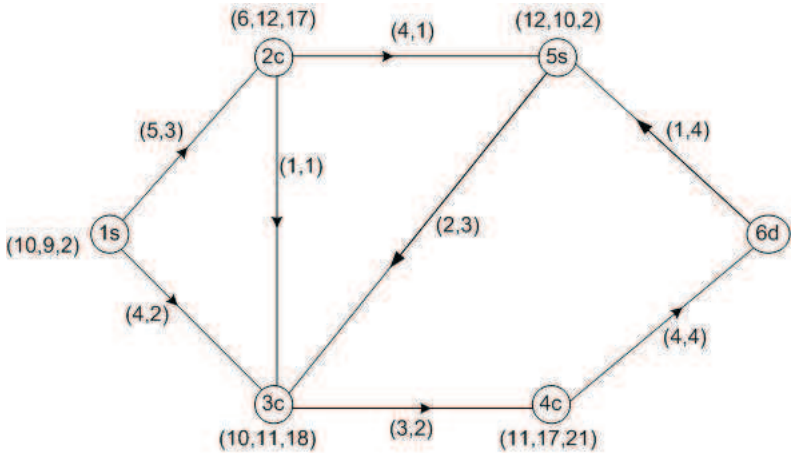

Figure 7.1: An illustration of the network topology.

are (28, 18, 1, 18, 4, 2, 1) and (38, 18, 2, 19, 4, 4, 2). Therefore, the optimal solution obtained by the algorithm is (28, 18, 1, 18, 4, 2, 1) and the corresponding path is $1 \to 3 \to 4 \to 6$ (labels: $(27, 10, 2, 19, -, -, 1) \to (21, 12, 1, 18, 1, 1, 2) \to (24, 14, 1, 18, 3, 2, 2) \to (28, 18, 1, 18, 4, 2, 1)$), total travel time is 8 and driver takes the passenger at 3c (see Fig. 7.2).

After that, we start with server 5s and the rest of passenger at (2c) and (4c) (see Fig. 7.3). Then, the optimal path is $5 \to 3 \to 4 \to 6$ (labels: $(17, 12, 2, 22, -, -, 1) \to (19, 15, 2, 22, 5, 1, 1) \to (11, 17, 1, 21, 3, 1, 2) \to (15, 21, 1, 21, 4, 2, 1)$), total travel time is 9 and driver takes the passenger at 4c.

At the destination node, we short all the labels $(c_i, t_i, -, -, -, -, -)$ by using R^2_+ lexicographic order (e.g., $(c_i, t_i, -, -, -, -, -) \leq (c_j, t_j, -, -, -, -, -)$ if and only if $c_i \leq c_j$ and $t_i \leq t_j$, and we said that label $(c_j, t_j, -, -, -, -, -)$ is dominant by label $(c_i, t_i, -, -, -, -, -)$), after we short and delete all non-dominant labels at the destination node. So, the set of dominant label at destination node is called $\mathcal{D}$. Now, we prove that if we have only one server s then (after terminal algorithm start from server s) $\mathcal{D}$ is the set of the efficient solution associated with the server s.

Theorem 7.1 *If there is only one server, then after terminal Algorithm 11, the set $\mathcal{D}$ is the set of the efficient solutions (i.e, the set $\mathcal{D}$ consists of all the Pareto points).*

Proof 7.1 *Firstly, assumption that there exists a label $L_j = (c_j, t_j, n_j, T_j, -, -, -)$ in $\mathcal{D}$ but it is not an efficient solution. So, there is an efficient path from the server node s to the destination node d such that its cost c_d and its departure time (at destination node) t_d satisfy $(c_j, t_j) > (c_d, t_d)$. Suppose this path is $P = (s \to v_{i_1} \to v_{i_2} \to \cdots \to v_{i_k} \to d)$, we calculate the labels $L_s \to L_{i_1} \to L_{i_2} \to \cdots \to L_{i_k} \to L_d$ corresponding with this path at each node from s to d. At the destination node d, the label is $L_d = (c_d, t_d, n_d, T_d, -, -, -)$. If the label L_d is belong to $\mathcal{D}$, so the label L_j is dominant by label L_d by R^2_+ lexicographic order, so L_j is already deleted at final step of Algorithm 11. It is a contradiction. Thus, label L_d is not belong to $\mathcal{D}$, then there is two possibilities:*

a) the label L_d is created at node d but it is deleted by an another dominant label L^*_d at destination node d, then L_j is dominated by the dominance label L^*_d. Then L_j is not belong to $\mathcal{D}$ and it is a contradiction.

b) in the path $P = (s \to v_{i_1} \to v_{i_2} \to \cdots \to v_{i_k} \to d)$, there is a node v_{i_e} where the label $L_{i_e} = (c_{i_e}, t_{i_e}, n_{i_e}, T_{i_e}, -, -, -)$ in path P is deleted by a dominant label $L^1_{i_e} =$

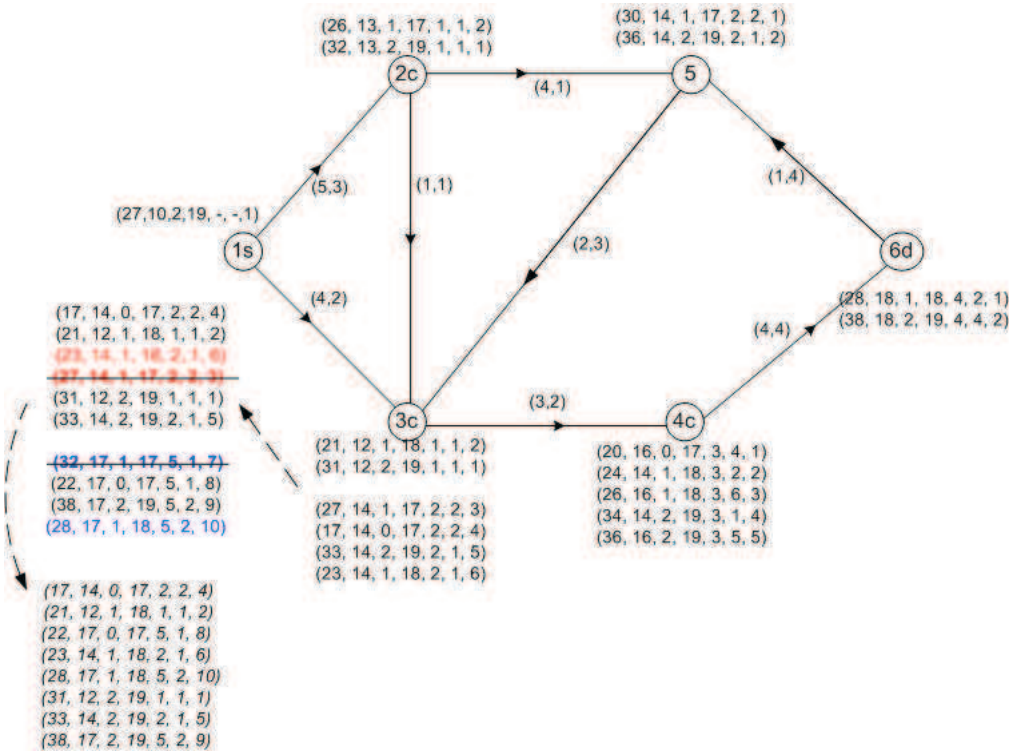

Figure 7.2: An illustration of our algorithm with driver 1s.

$(c^1_{i_e}, t^1_{i_e}, n^1_{i_e}, T^1_{i_e}, -, -, -)$ *(i.e.,* $c_{i_e} \geq c^1_{i_e}$, $t_{i_e} = t^1_{i_e}$, $n_{i_e} \leq n^1_{i_e}$, $T_{i_e} \leq T^1_{i_e}$*). Now, we consider a path* P_1 *which is created by the path from* s *to* v_{i_e} *corresponding with label* $L^1_{i_e}$ *and the rest of its path is belong to* P*. It is easy to show that* P_1 *is also an efficient path. We restart the proof with the path* P_1*. Because we have only finite paths from sever* s *to destination* d *then at last we obtain an efficient solution corresponding with a label* $L^*_d = (c_d, t_d, -, -, -, -, -)$ *at destination node* d*. Then label* L_j *is deleted by the dominant label* L^*_d*. It is also a contradiction.*

Secondly, we show that $\mathcal{D}$ *consists of all the efficient solution (all the Pareto points). Supposed that there exist an efficient solution which is not belong to* $\mathcal{D}$*. Then the efficient solution can be presented by a path* $P = (s \to v_{i_1} \to v_{i_2} \to \cdots \to v_{i_k} \to d)$*. By the similar way in the first step, the proof is consequence.*

7.4 Numerical results

The algorithm has been coded in VC++ and implemented on a Intel Core 2 CPU 2.53 Ghz, RAM 2GB.

We test our algorithm based on a real scenario, the network obtained from OpenStreetMap system (see Figure 7.4), which presents a part of Luxembourg city, Luxembourg. The scenario can be summarized as: we have a company and its employees, they want to go from their homes to workplace in the morning. The employees consist of drivers and passengers. The passengers want to participate with drivers whose cars have at least one available place to go to their workplace. The interval *[earliest departure time, latest arrival time]* of passengers is randomly generated in $[7h, 9h]$. The departure time of drivers are randomly generated in [7h,7h30]. The travel cost and travel time is generated based on the distance of its arc. The distance of one arc (u, v) is calculate by GPS position of nodes

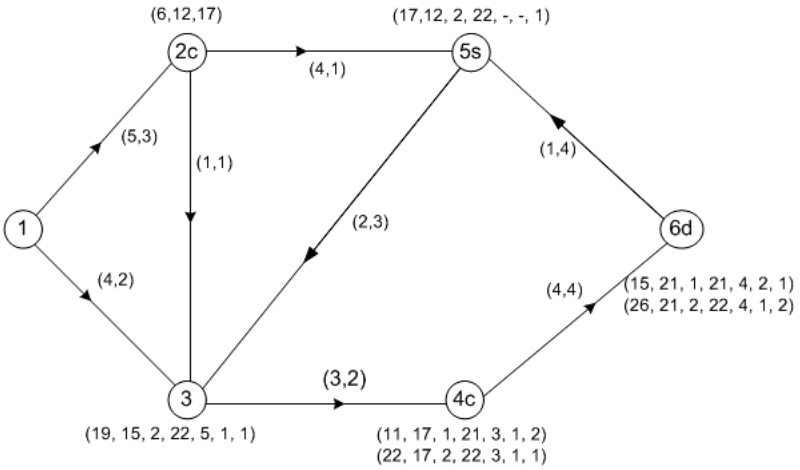

Figure 7.3: An illustration of our algorithm with driver 5s.

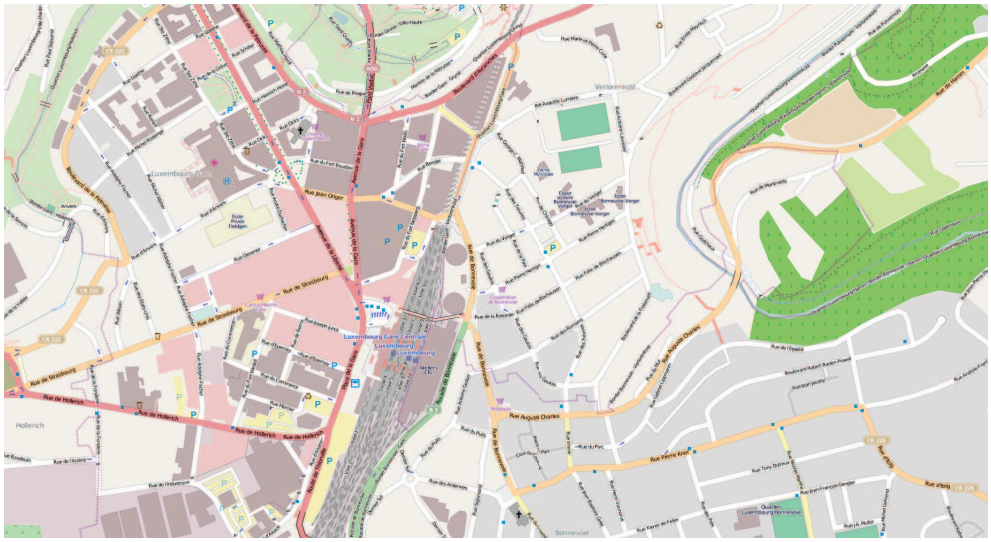

Figure 7.4: A part of Luxembourg city.

u and v. The latest arrival time of drivers (resp. penalty cost of passengers) is generated based on the shortest path from the driver (resp. the passenger) to the destination node. The number of available places is randomly generated in $\{2, 3, 4\}$. From the network in Figure 7.4, we consider the largest connected component, it can be expressed as a graph with 324 nodes and 918 arcs. We randomly generated the drivers and passengers and their corresponding parameters (i.e., departure time, latest arrival time,...). We consider ten datasets, which have pairs (number of driver, number of employees) = (81,162), (108,243), (120, 260), (121, 283), (129,259), (121, 288), (144,294), (147, 274), (202,288) and (140,300), respectively. For each dataset, we generated 15 sub-datasets which have the same number of drivers and number of employees, the other parameters are randomly generated as above, then we implement test our algorithm with each sub-dataset.

In Table 7.1, Num Servers, Num Employees, Ave-Total Time, Ave-Time per Server and Services (%) stand for the number of drivers, the number of employees, the average of total CPU time, the average of CPU time per driver (it equals to the average of total CPU time divide by the number of drivers) and the percentage of passengers who are serviced by drivers.

Num Servers	Num Employees	Ave-Total Time	Ave-Time per Server	Services (%)
81	162	212.22	2.62	66.67
108	243	362.97	2.69	68.89
120	260	376.94	2.92	73.84
121	283	414.28	3.42	70.98
129	259	380.59	2.92	69.92
121	288	457.64	3.78	68.26
144	294	483.13	3.36	79.33
147	274	437.05	2.97	71.65
202	288	578.06	2.86	68.60
140	300	438.50	3.13	69.38

Table 7.1: A numerical simulation of car pooling in a part of Luxembourg city.

From the numerical results, we observe that:

- The algorithm can give a set of efficient solution for each driver after a few second (2-3s).
- The percentage of passengers who are serviced of drivers is usually greater than 66%.
- For dynamic case, we can update new driver and new passenger after a few second. It can be applied in a real car pooling scenario.
- When the number of drivers and the number of passengers are changed but the original network is not changed, the average CPU times per driver are slight changed. It seem to be depend only the original network.

Algorithm complexity and Applications

In general, the algorithm has exponential complexity. The CPU time depend on the dimension (the size) and the structure, the density of the network. However, in the car pooling problem, we consider the road map, so the density of the network is low and the dimension also is not too large. As the results in Table 7.1 with the road map of Luxembourg city, we can get the results for each drivers after a few second. Thus, our algorithm is promising to apply in a real scenario with multiobjective and dynamic car pooling problem.

7.5 Conclusion

In this paper, we present a multiobjective car pooling problem. This is a "particular" problem in intelligent transportation system, as further as we know, there isn't any algorithm considering the car pooling problem in the form of the aforementioned problem. An efficient approach based on labeling algorithm is proposed for solving this problem. In general, the proposed algorithm can not guarantees the efficient solution in global context but it can solve rapidly the sub-problem (each step with one driver) then the set of efficient solution is found. Thus, users can choose the best solution depend on the priority of the criteria or the tradeoff between two criteria. Preliminary numerical simulations show that the algorithm is very interesting and promising to apply in real-time car pooling problems. Furthermore, the global context solution and the general multiobjective car pooling (e.g., multiple servers and multiple destinations) are also planing for our future work.

Conclusions and Perspectives

Conclusions

In this thesis, we have considered several optimization problems in Telecommunication, Mobility and Transport domain:

- Routing problems,
- Car pooling problems,
- Power control problems in wireless network,
- Optimal spectrum balancing problems in DSL networks.

Our methodologies focus on DC programming and DCA, Proximal decomposition method and Labeling method in graph theory. They are well-known as powerful tools in optimization. The considered problems were reformulated using the DC formulation/reformulation and exact penalty techniques, and the DCA was used to get a solution. Also, taking into account the structure of considered problems, we provide *appropriate* DC decompositions and relevant strategy of initial points for DCA in order to improve its performance. The main contribution is to develop new approaches for efficiently solving these problems, especially in very large dimension. Specifically:

- The formulation of nine problems in QoS routing, from the classical simple problems to the complex ones, is presented and casted into the form of Concave Quadratic Program or Binary Integer Linear Program. Then a solution method based on DC programming and DCA is investigated. In the Many to Many Multicast QoS routing problems, to handle a large number of constraints, we introduce the proximal decomposition technique in DCA to tackle convex subprograms at each iteration. We also provide some properties of the feasible set of these problems, which might be used to find *a good initial point* for the proposed algorithms. Numerical results obtained from the QoS routing problem solving are compared with CPLEX, the best solver for BILP.
- The mathematical formulation of Partitioning-Hub Location-Routing Problem (PHLRP) is analyzed and a new mathematical formulation of PHLRP is introduced as a Binary Integer Linear Program. To solve this problem, a new algorithm was proposed on the basis of DC programming and DCA. The preliminary numerical simulations are compared with CPLEX.
- A classical model of car pooling problem in the form of Mixed 0-1 Linear Programm was presented. The solution method, based on DC programming, DCA and combined DCA & Branch and Bound is explored. The numerical simulations of DCA, DCA-BB and CPLEX are compared.
- For Power control problems in wireless network and Optimal spectrum balancing problems, they are nonconvex and difficult to solve. An appropriate DC decomposition was proposed and well adapted for DCA. The numerical simulations are compared with those obtained from the well-known algorithms for solving these problems (GP, OSB, IWF, SIW, SCALE).

- A new multiobjective car pooling problem is introduced. It is followed by the study of a new distributed algorithm based on labeling algorithm for solving multiobjective shortest path problem. Preliminary numerical results in a real scenario with real data show the efficiency of our proposed algorithm and appear to be very promising in real time applications.

Moreover, the efficiency of our approaches was demonstrate by numerical results, the optimization algorithms were implemented in MATLAB, C/C++, in applications to many problems at the end of each chapter.

Perspectives and future works

- The DC algorithm designed and tested during this work provided us with interesting results in terms of solution accuracy and computational effort. However, theoretically, the computed solutions are still local optimal solutions. Recently, a new cutting technique developed for the DCA has allowed to reach global optimal solutions. We think that including such a promising technique in our next work would allow us to find the global optimal solutions for large-scale networks and thus obtain a decisive advantage compared with the other approaches.

- For car pooling algorithms, we may consider the case of multiple sources nodes (e.i, many cars and many passengers) and multiple destinations. Then we can plan to apply our algorithm in a real application, for instance, car pooling problem in the context of to-work problems.

- From the promising distributed algorithm to solve multiobjective car pooling, we plan to extend the algorithm for solving some multiobjective optimization problems in mobility transportation and try to find a better way to evaluate its solution.

- We also need to do further research inside DCA by more sophisticated development (e.g., the question of a *good* DC decomposition, an *appropriate* initial point for DCA and how to adapt this approach in case of a new problem to be solved).

Appendix

Appendix 1

A counterexample

In this section, we present an example which show that the constraints (3.5) are necessary (these constraint are missing in the Formulation 3 of [Ozsoy et al., 2008] and in the formulation of PHLRP of [Catanzaro et al., 2011]). We will show, by an example, that a solution of BILP exists with an couple of vertices (i, j) such that $w_{i,j} \neq w_{j,i}$. It is a contradiction with the definition of $w_{i,j}$.

Example 1

- The problem are generate with $m = 9,\ n = 26,\ \ell = 3,\ F_U = 4,\ F_L = 3, Y = 3$.
- The vertex set is $V = \{0, \cdots, 8\}$ and the set of arcs with the capacity in each arc are presented in the following:

$$\begin{bmatrix} C_{0,1} & C_{1,0} & C_{0,2} & C_{2,0} & C_{1,2} & C_{2,1} \\ C_{1,4} & C_{4,1} & C_{2,3} & C_{3,2} & C_{2,4} & C_{4,2} \\ C_{3,4} & C_{4,3} & C_{4,5} & C_{5,4} & C_{4,8} & C_{8,4} \\ C_{5,6} & C_{6,5} & C_{5,8} & C_{8,5} & C_{6,7} & C_{7,6} \\ C_{7,8} & C_{8,7} & & & & \end{bmatrix} = \begin{bmatrix} 95 & 71 & 119 & 152 & 64 & 80 \\ 200 & 30 & 166 & 30 & 20 & 150 \\ 127 & 66 & 57 & 49 & 64 & 16 \\ 31 & 21 & 54 & 22 & 94 & 76 \\ 141 & 70 & & & & \end{bmatrix}$$

- $T = \{(4, 2),\ (8, 6),\ (2, 8)\}$ and $d_{4,2} = 9,\ d_{8,6} = 1,\ d_{2,8} = 15$.
- The cost matrices are presented as follows $((c_{i,j}^{u,v}),\ (u, v) \in T,\ (i, j) \in A)$:

  –
  $$\begin{bmatrix} c_{0,1}^{4,2} & c_{1,0}^{4,2} & c_{0,2}^{4,2} & c_{2,0}^{4,2} & c_{1,2}^{4,2} & c_{2,1}^{4,2} \\ c_{1,4}^{4,2} & c_{4,1}^{4,2} & c_{2,3}^{4,2} & c_{3,2}^{4,2} & c_{2,4}^{4,2} & c_{4,2}^{4,2} \\ c_{3,4}^{4,2} & c_{4,3}^{4,2} & c_{4,5}^{4,2} & c_{5,4}^{4,2} & c_{4,8}^{4,2} & c_{8,4}^{4,2} \\ c_{5,6}^{4,2} & c_{6,5}^{4,2} & c_{5,8}^{4,2} & c_{8,5}^{4,2} & c_{6,7}^{4,2} & c_{7,6}^{4,2} \\ c_{7,8}^{4,2} & c_{8,7}^{4,2} & & & & \end{bmatrix} = \begin{bmatrix} 10 & 9 & 11 & 11 & 1 & 16 \\ 9 & 8 & 19 & 6 & 3 & 15 \\ 1 & 15 & 10 & 4 & 13 & 9 \\ 1 & 8 & 7 & 15 & 3 & 11 \\ 1 & 8 & & & & \end{bmatrix}$$

  –
  $$\begin{bmatrix} c_{0,1}^{8,6} & c_{1,0}^{8,6} & c_{0,2}^{8,6} & c_{2,0}^{8,6} & c_{1,2}^{8,6} & c_{2,1}^{8,6} \\ c_{1,4}^{8,6} & c_{4,1}^{8,6} & c_{2,3}^{8,6} & c_{3,2}^{8,6} & c_{2,4}^{8,6} & c_{4,2}^{8,6} \\ c_{3,4}^{8,6} & c_{4,3}^{8,6} & c_{4,5}^{8,6} & c_{5,4}^{8,6} & c_{4,8}^{8,6} & c_{8,4}^{8,6} \\ c_{5,6}^{8,6} & c_{6,5}^{8,6} & c_{5,8}^{8,6} & c_{8,5}^{8,6} & c_{6,7}^{8,6} & c_{7,6}^{8,6} \\ c_{7,8}^{8,6} & c_{8,7}^{8,6} & & & & \end{bmatrix} = \begin{bmatrix} 9 & 18 & 2 & 16 & 17 & 11 \\ 9 & 19 & 8 & 6 & 1 & 13 \\ 2 & 14 & 11 & 14 & 8 & 12 \\ 12 & 11 & 15 & 19 & 19 & 16 \\ 2 & 13 & & & & \end{bmatrix}$$

  –
  $$\begin{bmatrix} c_{0,1}^{2,8} & c_{1,0}^{2,8} & c_{0,2}^{2,8} & c_{2,0}^{2,8} & c_{1,2}^{2,8} & c_{2,1}^{2,8} \\ c_{1,4}^{2,8} & c_{4,1}^{2,8} & c_{2,3}^{2,8} & c_{3,2}^{2,8} & c_{2,4}^{2,8} & c_{4,2}^{2,8} \\ c_{3,4}^{2,8} & c_{4,3}^{2,8} & c_{4,5}^{2,8} & c_{5,4}^{2,8} & c_{4,8}^{2,8} & c_{8,4}^{2,8} \\ c_{5,6}^{2,8} & c_{6,5}^{2,8} & c_{5,8}^{2,8} & c_{8,5}^{2,8} & c_{6,7}^{2,8} & c_{7,6}^{2,8} \\ c_{7,8}^{2,8} & c_{8,7}^{2,8} & & & & \end{bmatrix} = \begin{bmatrix} 10 & 7 & 19 & 1 & 4 & 17 \\ 2 & 6 & 17 & 4 & 19 & 7 \\ 3 & 5 & 15 & 15 & 12 & 15 \\ 20 & 3 & 20 & 4 & 4 & 9 \\ 8 & 10 & & & & \end{bmatrix}$$

- The result obtained by CPLEX 9.0 of this problem is presented:
  $\phi_{1,2}^{4,2}=1,\ \phi_{4,1}^{4,2}=1,\ \phi_{7,6}^{8,6}=1,\ \phi_{8,7}^{8,6}=1,\ \phi_{0,1}^{2,8}=1,\ \phi_{2,0}^{2,8}=1,\ \phi_{1,4}^{2,8}=1,$
  $\phi_{4,8}^{2,8}=1,\ w_{0,1}=1,\ w_{0,2}=1,\ w_{0,4}=1,\ w_{1,2}=1,\ w_{1,4}=1,\ w_{2,4}=1,$
  $w_{2,7}=1,\ w_{2,8}=1,\ w_{3,6}=1,\ w_{3,7}=1,\ w_{4,7}=1,\ w_{4,8}=1,\ w_{5,0}=1,$
  $w_{5,2}=1,\ w_{6,2}=1,\ w_{6,7}=1,\ w_{7,2}=1,\ w_{7,8}=1,\ w_{8,3}=1,\ w_{8,6}=1,$
  $w_{8,7}=1,\ x_{0,0}=1,\ x_{1,0}=1,\ x_{2,0}=1,\ x_{3,7}=1,\ x_{4,0}=1,\ x_{5,5}=1,$
  $x_{6,7}=1,\ x_{7,7}=1,\ x_{8,7}=1,\ \tau_0^{1,4,2}=1$, and all the rest of variables are equal to 0.
- In this solution, we see that $w_{0,1}\neq w_{1,0},\ w_{0,2}\neq w_{2,0},\ w_{1,2}\neq w_{2,1}$.

Appendix 2

Solving Power control problem via Geometric programming

A. Geometric programming

This section presents a class of nonlinear optimization problems, in natural form, that is nonconvex problems. However, by using a change of variables and a transformation of the objective function and constraint functions, these problem can be transformed to convex optimization problems.

Monomial and posynomial function

A function $f: R^n \to R$ with $dom(f)=R_{++}^n$, defined as

$$f(x)=cx_1^{a_1}x_2^{a_2}\cdots x_n^{a_n} \tag{7.5}$$

where $c>0$ and $a_i\in R$, is called a *monomial function* (or a monomial[1]). A sum of monomials is called a *posynomial function* (or a posynomial) with K terms, i.e., a function in the form of

$$f(x)=\sum_{k=1}^{K} c_k x_1^{a_{1k}}x_2^{a_{2k}}\cdots x_n^{a_{nk}} \tag{7.6}$$

where $c_k>0$.

Geometric programming problem

An optimization problem of the form

$$\begin{aligned} &\text{minimize} && f_0(x)\\ &\text{subject to} && f_i(x)\leq 1;\ i=1,\cdots,m.\\ &&& h_i(x)=1;\ i=1,\cdots,p. \end{aligned}$$

where $f_0,\cdots,f_m$ are posynomials and $h_1,\cdots,h_p$ are monomials, is called a *geometric program (GP)*. The domain of this problem is $D=R_{++}^n$; the constraint $x\succ 0$ is implicit.

In general, geometric programs are *not convex optimization problems*, nonetheless, by a change of variables and a transformation of the objective and constraint functions, they can be transformed to *convex problems*.

The variables y_i is defined as $y_i=\log(x_i)$, so $x_i=e^{y_i}$. So, if f is the monomial function of x, i.e.,

$$f(x)=\sum_{k=1}^{K} c_k x_1^{a_{1k}}x_2^{a_{2k}}\cdots x_n^{a_{nk}}$$

[1]The term 'monomial' conflicts with the standard definition from algebra, in which the exponents must be nonnegative integers, but this should not cause any confusion.

then

$$\begin{aligned} f(x) &= f(e^{y_1}, \cdots, e^{y_n}) \\ &= c(e^{y_1})^{a_1} \cdots (e^{y_n})^{a_n} \\ &= e^{a^T y + b}, \end{aligned}$$

where $b = \log(c)$.

Similarly, if f is the posynomial (7.6), i.e.,

$$f(x) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} \cdots x_n^{a_{nk}}$$

then

$$f(x) = \sum_{k=1}^{K} e^{a_k^T y + b_k}$$

where $a_k = (a_{1k}, \cdots, a_{nk})$ and $b_k = \log(c_k)$.

The geometric program in terms of the new variable y can be described in the following,

$$\begin{aligned} \text{minimize} \quad & \sum_{k=1}^{K_0} e^{a_{0k}^T y + b_{0k}} \\ \text{subject to} \quad & \sum_{k=1}^{K_i} e^{a_{ik}^T y + b_{ik}} \leq 1; \ i = 1, \cdots, m, \\ & e^{g_i^T y + h_i} = 1; \ i = 1, \cdots, p, \end{aligned}$$

where $a_{ik} \in R^n$, $i = 0, \cdots, m$, contain the exponents of the posynomial inequality constraints, and $g_i \in R^n$, $i = 1, \cdots, p$, contain the exponents of the monomial equality constraints of the original geometric program.

Using the logarithm function to transform the objective and constraint functions, so this results in the problem

$$\begin{aligned} \text{minimize} \quad & \widetilde{f}_0(y) = \log(\sum_{k=1}^{K_0} e^{a_{0k}^T y + b_{0k}}) \\ \text{subject to} \quad & \widetilde{f}_i(y) = \log(\sum_{k=1}^{K_i} e^{a_{ik}^T y + b_{ik}}) \leq 0; \ i = 1, \cdots, m, \\ & \widetilde{h}_i(y) = g_i^T y + h_i = 0; \ i = 1, \cdots, p. \end{aligned}$$

Because the functions $\widetilde{f}_i$ are convex ([Boyd and Vandenberghe, 2003]), and $\widetilde{h}_i$ are affine, this problem is a *convex optimization problem*.

We see that the transformation between the posynomial form geometric program and the convex form geometric program does not consists of any computation, the problem data for the two problems are the same. It only changes the form of the objective and constraint functions.

B. Solving Power control problem via Geometric programming

We consider the context of high-SINR. Prior work usually made use of a high-SINR approximation, i.e.,

$$\mathrm{R}_k = \log_2(1 + \mathrm{SINR_k}) \approx \log_2(\mathrm{SINR_k}) \tag{7.7}$$

on the link rates. We have

$$\begin{aligned}\mathrm{R}_k &\approx \log_2(\mathrm{SINR_k}) \\ &\approx \log_2(\frac{L\overline{G}_k P_k}{\sum_{j\neq k}^{K}\overline{G}_j P_j + \sigma_k^2}) \\ &\approx \log_2(L\overline{G}_k) - \log_2(\sum_{j\neq k}^{K}\overline{G}_j P_j P_k^{-1} + \sigma_k^2 P_k^{-1}).\end{aligned}$$

We replace R_k by $\log_2(L\overline{G}_k) - \log_2(\sum_{j\neq k}^{K}\overline{G}_j P_j P_k^{-1} + \sigma_k^2 P_k^{-1})$ and P_l by variable transformation $\widetilde{P}_l = \log(P_l)$ in Problem 2. It is easy to see that it is a geometric programming in the form of convex programming. So, we can use an efficiently available software for solving GP problem such as CVX that are using in Chapter 4.

Appendix 3

Successive Convex Approximation for Low complExity (SCALE)

Successive Convex Approximation for Low complExity (SCALE) was first introduced in [Papandriopoulos and Evans, 2006] and [Papandriopoulos and Evans, 2009]. It solves problem (5.3) by approximating it with a concave lower bound (to avoid the d.c. structure), maximizing the approximation, this is a convex optimization problem, and repeating the process with another approximation. SCALE introduces a method of distributing the required processing over multiple users.

SCALE users the lower bound approximation in the following

$$\alpha\log(z) + \beta \leq \log(1+z) \tag{7.8}$$

where $\alpha = \frac{z_0}{1+z_0}$ and $\beta = \log(1+z_0) - \frac{z_0}{1+z_0}\log(z_0)$. The inequality is tight when $z = z_0$.

Applying (7.8) to the optimization problem (5.3), results in the relaxation problem is[2]

$$\max \sum_n \sum_k w_n(\alpha_k^n \log(\frac{1}{\Gamma}\frac{p_k^n}{\sum\limits_{m\neq n} h_k^{n,m} p_k^m + \sigma_k^n}) + \beta_k^n) \tag{7.9}$$

$$\begin{aligned}\text{s.t.}\quad &\sum_k p_k^n \leq P_n\ \forall n, \\ &0 \leq p_k^n \leq p_k^{n,mask}\ \forall k,n.\end{aligned}$$

Let $\widetilde{p}_k^n = \log(p_k^n)$, so the new problem is in the form

$$\max \sum_n \sum_k w_n(\alpha_k^n \log(\frac{1}{\Gamma}e^{\widetilde{p}_k^n}) - \alpha_k^n \log(\sum_{m\neq n} h_k^{n,m} e^{\widetilde{p}_k^m} + \sigma_k^n) + \beta_k^n) \tag{7.10}$$

$$\begin{aligned}\text{s.t.}\quad &\sum_k e^{\widetilde{p}_k^n} \leq e^{\widetilde{P}_n}\ \forall n, \\ &0 \leq e^{\widetilde{p}_k^n} \leq e^{\widetilde{p}_k^{n,mask}}\ \forall k,n.\end{aligned}$$

The bound of the achievable system rate is improved periodically by maximizing a lower-bound. The results of the procedure is presented below. The initial choice of the constants $\{\alpha,\beta\}$ is not critical, it makes use of a simple high-SINR approximation with $\alpha = 1$ and $\beta = 0$.

[2] We replace $\log_2(.)$ by $\log(.)$ for simple regarding. This replacement is not effect to the computation, it only changes the value of objective function by divide $\log 2$.

1. initialize iteration counter $t = 0$
2. initialize all $\alpha_k^n(t) = 1,\ \beta_k^n(t) = 0$ (high-SINR approx.)
3. **repeat**
4. maximize: solve sub-problem (7.10) to give solution $p_k^n(t)$
5. tighten: update $\alpha_k^n(t+1)$ and $\beta_k^n(t+1)$ at $z_0(p_k^n(t))$
6. increment t
7. **until** convergence

Note that, the sequence of iterates produces a monotonically increasing objective and will always converge (see [Papandriopoulos and Evans, 2006]).

Any convex optimization tool can be used to solve Problem (7.10), but SCALE proposes a gradient approach that can also be distributed among each users by using the function's Lagrangian:

$$\mathcal{L}(\widetilde{P}, \lambda) = \sum_n \sum_k w_n(\alpha_k^n \log(\frac{1}{\Gamma} e^{\widetilde{p}_k^n}) - \alpha_k^n \log(\sum_{m \neq n} h_k^{n,m} e^{\widetilde{p}_k^m} + \sigma_k^n) + \beta_k^n) - \sum_n \lambda_n (\sum_k e^{\widetilde{p}_k^n} - P_n). \tag{7.11}$$

The corresponding dual problem is then $\min_{\lambda} \max_{\widetilde{P}} \mathcal{L}(\widetilde{P}, \lambda)$.

We update dual variables λ_n through a gradient descent

$$\lambda_n^{(t+1)} = \max\{0,\ \lambda_n^{(t)} + \varepsilon(\sum_k e^{\widetilde{p}_k^n(t)} - P_n)\} \tag{7.12}$$

for fixed P_k, where ε is a sufficiently small step-size and t is an iteration number for the sub-problem. Each λ_n is updated locally by each user n.

By taking the Lagrangian's gradient with respect $\widetilde{P}$ (and λ fixed) then setting it equal to zero, the spectrum update formula is obtained:

$$p_k^n = \frac{w_n \alpha_k^n}{\lambda_n + \sum_{m \neq n} \frac{h_k^{m,n} w_m \alpha_k^m}{\sum_{q \neq m} h_k^{m,q} p_k^q + \sigma_k^m}}. \tag{7.13}$$

The denominator of the update equation, Equation (7.13), contains information about other users. This additional information allows the distributed algorithm to converge to a locally optimal point. In order to gather this information, a message passing system is required. Every user measures their total interference and noise on every tone and transmits it back to the Spectrum Management Center (SMC). Since the SMC has partial channel knowledge, the message passing system and update formula can be simplified as follows:

$$\mathcal{N}_k^n = \frac{w_n \alpha_k^n}{\sum_{m \neq n} h_k^{n,m} p_k^m + \sigma_k^n} \tag{7.14}$$

$$\mathcal{M}_k^n = \sum_{m \neq n} h_k^{m,n} \mathcal{N}_k^m \tag{7.15}$$

$$p_k^n = \frac{w_n \alpha_k^n}{\lambda_n + \mathcal{M}_k^n} \tag{7.16}$$

At every iteration, every user n calculates N_k^n on every tone and sends it to the SMC. The SMC produces the M_k^n values and distributes them to each user n. The full SCALE algorithm is summarized in SCALE Algorithm below.

SCALE Algorithm

At each user n's modem:
Initialize PSD: $s_k^n = 0,\ \forall k$;
Initialize $\alpha_k^n = 1,\ \forall k$;
repeat
    Receive $\mathcal{M}_k^n$ from SMC;
    Update spectrum using (17);
    At every m iterations, update $\alpha_k^n, \forall k$;
    Generate $\mathcal{N}_k^m$ and send to SMC ;
indefinitely
At the SMC:
repeat
    Receive $\mathcal{N}_k^m$ from every user;
    Generate M_k^m and send to SMC;
indefinitely

Index

References

N. B. Ali, M. Molnar, and A. Belghith. Multi-constrained qos multicast routing optimization. Public interner 1882, IRISA, 2008.

S. Alumur and B. Y. Kara. Network hub location problems: The state of the art. *European Journal of Operational Research*, 190(1):1–21, 2008.

A. Auslender. *Optimisation Méthodes Numériques*. Paris: Masson, 1976.

R. Baldacci, V. Maniezzo, and A. Mingozzi. An exact method for the car pooling problem based on lagrangean column generation. *Oper. Res.*, 52(3):422–439, 2004.

F. Bauer. *Multicast routing in point-to-point networks under constraints*. PhD dissertation, University of Carlifornia, Santa Cruz, 1996.

M. Biguesh, S. Shahbazpanahi, and A. B. Gershman. Robust downlink power control in wireless cellular systems. *EURASIP J. Wireless Communications and Networking, special issue on Multiuser MIMO Networks*, 2:261–272, 2004.

D. Blokh and G. Gutin. An approximation algorithm for combinatorial optimization problems with two parameters. *AUSTRALASIAN J. COMBIN*, 14:157–164, 1995.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2003.

R. W. Calvo, F. L. Luigi, P. Haastrup, and V. Maniezzo. A distributed geographic information system for the daily car pooling problem. *Computers and Operations Research*, 31:2263–2278, 2004.

J. F. Campbell. *Strategic network design for motor carriers*. Springer, U.S.A., 2005.

D. Catanzaro, E. Gourdin, M. Labbe, and F. A. Ozsoy. A branch-and-cut algorithm for the partitioning-hub location-routing problem. *Computers & Operations Research*, 38 (2):539–549, 2011.

R. Cendrillon and M. Moonen. Iterative spectrum balancing for digital subscriber lines. In *Proc. IEEE ICC2005*, Seoul, Korea, May 2005.

R. Cendrillon, W. Yu, M. Moonen, J. Verlinden, and T. Bostoen. Optimal multiuser spectrum management for digital subscriber lines. *IEEE Trans. Comm.*, 54:922–933, May 2006.

R. Cendrillon, J. W. Huang, M. Chiang, and M. Moonen. Autonomous spectrum balancing for digital subscriber lines. *IEEE Trans. Signal. Processing*, 55(8):4241–4257, 2007.

S. Chen and C. Nahrstedt. On finding multi-constrained paths. *International Journal of Computational Geometry and Applications*, 1998.

M. Chiang. Geometric programming for communication systems. *Foundations and Trends in Communications and Information Theory*, 2:1–154, 2005.

M. Chiang. *Nonconvex Optimization for Communication Networks*, volume 3 of *Advances in Mechanics and Mathematics*, chapter 5. Dedicated to Gilbert Strang on the Occasion of His 70th Birthday. Edited by David Y. Gao & Hanif D. Sherali., 2006a.

M. Chiang. Chapter 5: Nonconvex optimization for communication networks. *Advances in Mechanics and Mathematics*, 3:136–196, 2006b.

M. Chiang, P. Hande, T. Lan, and C. W. Tan. Power control in wireless cellular networks. *Foundation and Trends in Networking*, 2:1–156, 2008.

J. F. Cordeau and G. Laporte. The dial-a-ride problem (darp): Variants modeling issues and algorithms. *4OR*, 1:89–101, 2003.

J. F. Cordeau and G. Laporte. The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, 153:29–46, 2007.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithm.* MIT Press, 1997.

M. Ehrgott. *Multicriteria Optimization.* Lecture Notes in Economics and Mathematical Systems. Second edition, 2005.

F. Rashid Farrokhi, K. J. R. Liu, and L. Tassiulas. Transmit beamforming and power control for cellular wireless systems. *IEEE Journal Selected Areas in Communicatiopns*, 16:1437–1450, 1998.

G. J. Foschini and Z. Miljanic. A simple distributed autonomous power control algorithm and its convergence. *IEEE Trans. Veh. Technol.*, 42:641–646, 1993.

R. Gallager, P. Humblet, and P. Spira. A distribited algorithm for minimum-weight spanning tree. *ACM Transactions an Programming Languages and Systems*, pages 66–77, 1983.

M. Grant and S. Boyd. *CVX: Matlab Software for Disciplined Convex Programming.* Version 1.21 (Build 781), http://cvxr.com/cvx/download, 2010.

M. Grotschel and Y. Wakabayashi. Facets of the clique partitioning polytope. *Mathematical Programming: Series A and B*, 47(3):367–387, 1990.

R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, 1992.

R. Horst and Hoang Tuy. *Global optimization (Deterministic Approaches).* Springer-Verlag, Berlin, third edition, 1996.

R. Horst, P. M. Pardalos, and N. V. Thoai. *Introduction to global optimization.* Dordrecht, Netherlands: Kluwer Academic Publishers, 1995.

F.K. Hwang. *Steiner problems*, pages 55–89. Networks. 1992.

D. Julian, M. Chiang, D. O'Neill, and S. P. Boyd. Qos and fairness constrained convex optimization of resource allocation for wireless cellular and ad hoc networks. *Proc. IEEE INFOCOM'02*, 1:477–486, 2002.

S. Kandukuri and S. P. Boyd. Optimal power control in interference-limited fading wireless channels with outage-probability specifications. *IEEE Trans. Wireless Communications*, 1:46–55, 2002.

T. Korkmaz and M. Krunz. Multi-constrained optimal path selection. In *Proceedings IEEE INFOCOM 2001*, volume 2, pages 834–843, 2001.

T. Korkmaz, M. Krunz, and S. Tragoudas. An efficient algorithm for finding a path subject to two additive constraints. *Computer Communications Journal*, 25(3):225–238, 2002.

F. Kuipers and P. V. Mieghem. Mamcra: A constrained-based multicast routing algorithm. *Computer Communications*, 25(8):801–810, 2002.

F. A. Kuipers and Piet F. A. Van Mieghem. Conditions that impact the complexity of qos routing. *Networking, IEEE/ACM Transactions*, 13, 2005.

F. A. Kuipers, T. Korkmaz, M. Krunz, and P. Van Mieghem. Overview of constraint-based path selection algorithms for qos routing. *IEEE Commun. Mag.*, 40(12):50–55, 2002.

P. J Laurent. *Approximation et optimisation.* Paris: Hermann, 1972.

H. A Le Thi. *Analyse numérique des algorithmes de l'optimisation DC. Approches locale et globale. Codes et simulations numériques en grande dimension. Applications.* Thèse de doctorat, Université de Rouen, 1994.

H. A Le Thi. *Contribution à l'optimisation non convexe et l'optimisation globale: Théorie, Algorithmes et Applications.* Habilitation à diriger des recherches, Université de Rouen, 1997.

H. A Le Thi and T. Pham Dinh. Solving a class of linearly constrained indefinite quadratic problems by dc algorithms. *Journal of Global Optimization*, 11:253–285, 1997.

H. A Le Thi and T. Pham Dinh. A continuous approach for globally solving linearly constrained quadratic zero - one programming problems. *Optimization*, 50:93–120, 2001.

H. A Le Thi and T. Pham Dinh. The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems. *Annals of Operations Research*, 133:23–46, 2005.

H. A Le Thi, T. Pham Dinh, and M. Le Dung. Exact penalty in dc programming. *Vietnam Journal of Mathematics*, 27, 1999.

H. A Le Thi, T. Pham Dinh, and T. Nguyen Van. Combination between local and global methods for solving an optimization problem over the efficient set. *European Journal of Operational Research*, 142:257–270, 2002.

H. A. Le Thi, T. P Nguyen, and T. Pham Dinh. A continuous dc programming approach to the strategic supply chain design problem from qualified partner set. *European Journal of Operational Research*, 183, 2007.

H. A. Le Thi, N. B Mbaye, and T. Pham Dinh. Solving a multimodal transport problem by dc. In *Proc. IEEE International conference on Research, Innovation and Vision for the future in Computing & Communications Technologies*, pages 49–56, Ho Chi Minh city, Vietnam, 2008. IEEEXplore.

H. A. Le Thi and T. Pham Dinh. D.c. programming approaches for multicommodity network optimization problems with step increasing cost functions. *Journal of Global Optimization*, 22:204–233, 2002.

H. A. Le Thi, Q. T. Nguyen, T. Pham Dinh, and T. K. Phan. Energy minimization-based cross-layer design in wireless networks. In *Proceedings of the 2008 High Performance Computing & Simulation Conference (HPCS 2008)*, pages 283–289, Nicosia, Cyprus, 2008a.

H. A. Le Thi, Q. T. Nguyen, T. Pham Dinh, and T. K. Phan. Cross-layer optimization in multi-hop tdma networks using dca. In *17th International Conference on Computer Communications and Networks (ICCCN 2008)*, 2008b.

G. Liu and K. G. Ramakrushnam. A*prune: an algorithm for finding k shortest paths subject to multiple constraints. In *Proceedings of IEEE INFOCOM*, volume 2, pages 743–749, 2001.

D. H. Lorenz and Raz. Danny. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28(5):213–219, 2001.

D.H. Lorenz, A. Orda, D. Raz, and Y. Shavitt. Efficient qos partition and routing of unicast and multicast. In *Proceedings IEEE/IFIP IWQoS, Pittsburgh, PA*, 2000.

R. Lui and W. Yu. Low-complexity near-optimal spectrum balancing for digital subscriber lines. In *Proc. IEEE ICC2005*, Seoul, Korea, May 2005.

P. Mahey, S. Oualibouch, and T. Pham Dinh. Proximal decomposition on the graph of a maximal monotone operator. *SIAM J. Optim.*, 5:454–466, May 1995.

K. McCammon. G. vdsl: Vdsl band plan for north america. Technical report, ITU, ITU contribution D. 715, 2000.

P. V. Mieghem and F. A. Kuipers. On the complexity of qos routing. *Computer Communications*, 26(4):376–387, 2003.

P. V. Mieghem and F. A. Kuipers. Concepts of exact qos routing algorithms. *IEEE/ACM Trans. on Networking*, 12(5):851–864, 2004.

J. J. Moreau. Proximité et dualité dans un espace hibertien. *Bull. Soc. Math. France*, 93: 273–299, 1965.

V. Oksman and J. M. Cioffi. Noise models for vdsl performance verification. Technical report, ANSI, ANSI –T1E1.4/99-438R2, 1999.

A. Orda and A. Sprintson. Efficient algorithm for computing disjoint qos paths. In *Proceedings of IEEE INFOCOM*, volume 1, pages 727–738, 2004.

F. A. Ozsoy, M. Labbe, and E. Gourdin. Analytical and empirical comparison of integer programming formulations for a partitioning-hub location-routing problem. Technical Report 586, Fanstord University, ULB, Department of Computer Science, 2008.

J. Papandriopoulos and J. S. Evans. Low-complexity distributed algorithms for spectrum balancing in multi-user dsl networks. In *Proc. IEEE ICC2006*, Istanbul, Turkey, 2006.

J. Papandriopoulos and J. S. Evans. Scale: A low-complexity distributed protocol for spectrum balancing in multiuser dsl networks. *IEEE Trans. Inform. Theory*, 55:3711–3724, 2009.

S. N. Parragh, K. F. Doerner, and R. F. Hartl. A survey on pickup and delivery problems, part ii: Transportation between pickup and delivery locations. *Journal fur Betriebswirtschaft*, 58(2):81–117, 2008.

Pragyansmita Paul and S. V. Raghavan. Survey of multicast routing algorithms and protocols. In *Proceedings of the 15th international conference on Computer communication (ICCC 02)*, Washington DC, USA, 2002.

T. Pham Dinh. Elements homoduaux relatifs à un couple de normes (φ, ψ). applications au calcul de $s_{\varphi\psi}(a)$. Technical report, Grenoble, 1975.

T. Pham Dinh. Calcul du maximum d'une forme quadratique définie positive sur la boule unité de la norme du max. Technical report, Grenoble, 1976.

T. Pham Dinh. *Algorithms for solving a class of non convex optimization problems. Methods of subgradients*. Fermat days 85. Mathematics for Optimization. Elsevier Science Publishers B.V. North-Holland, 1986.

T. Pham Dinh. *Duality in DC (difference of convex functions) optimization. Subgradient methods*, volume 84 of *Trends in Mathematical Optimization, International Series of Numer Math.*, pages 277–293. 1988.

T. Pham Dinh and H. A Le Thi. Dc optimization algorithms for solving the trust region subproblem. *SIAM Journal of Optimization*, 8(2):476–505, 1998.

T. Pham Dinh, N. Nguyen Canh, and H. A Le Thi. An efficient combined dca and b&b using dc/sdp relaxation for globally solving binary quadratic programs. *J. Global Optimization*, 48(4):595–632, 2010.

T. Pham Dinh and H. A. Le Thi. Convex analysis approach to dc programming: Theory, algorithms and applications. *Acta Mathematica Vietnamica*, 22:289–357, 1997.

DSM Report. Ansi nipp-nai contribution 2005-031r5. Technical report, Las Vegas, NV, 2005.

R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, 1970.

R. T. Rockafellar. Monotone operator and the proximal point algorithm. *SIAM J. control and optimization*, 14(5):877–898, 1976.

S. Ropke, J. F. Cordeau, and G. Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49:258–272, 2007.

H.F. Salama. *Multicast routing for real time communication in high speed networks*. PhD dissertation, North Calorina State University, Raleigh, 1996.

A. Schrijver. *Theory of Linear and Integer Programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Second edition, 1998.

K. B. Song, S. T. Cheung, G. Ginis, and J. M. Cioffi. Dynamic spectrum management for next-generation dsl systems. *IEEE Comm. Mag.*, 40:101–109, 2002.

J. E. Spingarn. Partial inverse of a monotone operator. *Applied Mathematics and Optimization*, 10(1):247–265, 1983.

T. Starr, J. M. Cioffi, and P. Silverman. *Understanding Digital Subscriber Line Technology*. Prentice Hall, 1999.

A. S Ta, H. A Le Thi, D. Khadraoui, and T. Pham Dinh. Solving qos routing problems by dca. In *Proc. 2th ACIIDS, Intelligent Information and Database Systems, Lecture Notes in Artificial Intelligence (LNAI)*, pages 460–470, Hue, Vietnam, 2010a. Springer Verlag 5991.

A. S Ta, H. A Le Thi, D. Khadraoui, and T. Pham Dinh. Solving multicast qos routing problem in the context v2i communication services using dca. In *Proc. 9th IEEE/ACIS (ICIS 2010)*, pages 471–476, Yamagata, Japan, 2010b. IEEEXplore.

A. S Ta, H. A Le Thi, G. Arnould, D. Khadraoui, and T. Pham Dinh. Solving car pooling problem using dca. In *Proc. Global Information Infrastructure Symposium (GIIS)*, pages 1–6, Danang, Vietnam, 2011. IEEEXplore.

A. S Ta, H. A Le Thi, D. Khadraoui, and T. Pham Dinh. Solving partitioning-hub location-routing problem using dca. *Journal of Industrial and Management Optimization (JIMO)*, 8(1):87–102, 2012a.

A. S Ta, H. A Le Thi, T. Pham Dinh, and D. Khadraoui. A distributed algorithm solving multiobjective dynamic car pooling problem. In *Proc. International Conference on Computer and Informatic Science*, Kuala Lumpur, Malaysia, 2012b.

A. S Ta, H. A Le Thi, T. Pham Dinh, and D. Khadraoui. Solving many to many multicast qos routing problem using dca and proximal decomposition technique. In *Proc. International Conference on Computing, Networking and Communications 2012*, pages 809–814, Hawaii, American, 2012c. IEEEXplore.

J. F Toland. Direct calculation of the information matrix via the EM algorithm. *Journal of Mathematical Analysis and Applications*, 66:399–415, 1978.

Hoang Tuy. *Global Optimization : Deterministic Approaches*. Springer-Verlag, Berlin, second edition, 1993.

Hoang Tuy. *DC Optimisation : Theory, Methods and Algorithms, Handbook of Global Optimisation*, pages 149–216. Horst and Pardalos eds, Klaver Academic Publishers, 1995.

Hung-Ying Tyan, Jennifer C. Hou, and Bin Wang. Many-to-many multicast routing with temporal quality of service guarantees. *IEEE Transaction on Computers*, 52(6):826–832, 2003.

J. B. H Urruty. *Generalized differentiability, duality and optimization for problem dealing with differences of convex functions*, volume 256 of *Lecture Notes in Economics and Mathematical Systems*, pages 260–277. Springer Verlag, 1985.

J. B. H Urruty. *Conditions nécessaires et suffisantes d'optimalité globale en optimisation de différences de deux fonctions convexes*, pages 459–462. I. CRAS, 1989.

B. Wang and J. C. Hou. Multicast routing and its qos extention: Problems, algorithms and protocols. *Network, IEEE*, 14(1):22–36, 2000.

Q. Warburton. Approximation of pareto optima in multiple objective shortest path problems. *Operations Research*, 35:70–79, 1987.

Y. Xu, S. Panigrahi, and T. Le Ngoc. Selective iterative water filling for digital subscriber lines (dsl). *EURASIP J. Appl. Signal Process.*, 2007, 2007.

R. D. Yates. A framework for uplink power control in cellular radio systems. *IEEE Journal Selected Areas in Communications*, 13:1341–1347, 1995.

W. Yu, G. Ginis, and J. M. Cioffi. Distributed multiuser power control for digital subscriber lines. *IEEE JSAC*, 20:1105–1115, 2002.

X. Yuan and X. Liu. Heuristic algorithms for multi-constrained quality of service routing. In *Proceedings of IEEE INFOCOM*, volume 2, pages 844–853, 2001.

L. Zhang, L. Cai, M. Li, and F. Wang. A method for least-cost qos multicast routing based on genetic simulated annealing algorithm. *Computer Communications*, 32:105–110, 2009.