TECHNICAL UNIVERSITY OF CRETE
ELECTRONIC AND COMPUTER ENGINEERING DEPARTMENT
TELECOMMUNICATIONS DIVISION

Decomposition Methods for Network Utility Maximization

by

Giorgos Kostoulas

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DIPLOMA DEGREE OF

ELECTRONIC AND COMPUTER ENGINEERING

March 2015

THESIS COMMITTEE

Professor Athanasios P. Liavas, *Thesis Supervisor*
Professor Michael Paterakis
Associate Professor Polychronis Koutsakis

Abstract

Network Utility Maximization (NUM) is the problem of allocating the right amount of resources to the nodes of a network, in order to maximize an overall utility function. There are many optimization tools to solve this problem in a centralized manner. In this thesis, we discuss distributed ways to solve various formulations of NUM problems. We decompose the problems into subproblems using Primal Decomposition, by applying direct resource allocation and then adjust the resources by small steps until equilibrium, and Dual Decomposition by pricing the resource in such manner that each node achieves the optimal utility. Many alternatives can be derived from these two methods, in different NUM formulations, with the use of multilevel decompositions. These decompositions may lead to better understanding of existing networks, reverse engineering of network protocols like TCP, better management of existing networks, and ways to design and operate new networks by layering as optimization. Finally, we experiment with the message passing of these algorithms and try to minimize the data transferred by quantizing the values.

Acknowledgements

I would like to thank everyone that helped me with this thesis.

I would like to express my gratitude to Professor Athanasios Liavas, my thesis supervisor, for giving me the chance to work on this thesis, for his patient guidance, and for his useful critiques on this work.

I would like to thank my friends for the adorable moments I had with them and for their support.

I would like to offer my special thanks to my sister Katerina and my brother Nikos for the guidance and valuable support during my student life.

Finally, I wish to thank my parents for their support and encouragement throughout my study.

Table of Contents

List of Figures

List of Abbreviations

NUM	Network Utility Maximization
KKT	Karush-Kuhn-Tucker
LP	Linear Programming
QoS	Quality-of-Service
TDMA	Time Division Multiple Access
FDMA	Frequency Division Multiple Access
TCP	Transmission Control Protocol
XCP	eXplicit Control Protocol
RCP	Rate Control Protocol

Chapter 1

Introduction

1.1 Motivation

The basic idea of decomposition methods is to solve a problem by solving smaller subproblems coordinated by a master problem. The distributed way to solve problems that probably cannot be solved in a centralized way, for various reasons, such as privacy or memory issues, has limitless applications. The study of these methods for various Network Utility Maximization (NUM) problems leads to the most appropriate distributed algorithm for a given network resource allocation problem, and it quantifies the comparison across architectural alternatives of distributed, layered network control. The work of Daniel P. Palomar and M.Chiang [1] and [2] was the main motivation for this thesis and a great help for understanding these methods. The growing interest in these algorithms and their extensions have made many of the world top Universities to add into their optimization courses these techniques.

Our goal is to present the decomposition methods so that the reader will be able to understand and apply these algorithms to a problem he is facing. This is achieved by presenting the framework and many different examples in network problem formulations.

In engineering, the best way to understand a complex problem is to break it down into simpler problems. Studying decomposition methods in networks is the best way to understand the problems of resource allocation and functionality allocation and then to try to obtain the most appropriate algorithm to solve a given problem. Perhaps, even more importantly, it quantifies the comparison across architectural alternatives of modularized network design. A paramount issue in the design of network architecture is where to place functionalities and how to connect them, an issue that is often more critical than the detailed design of how to carry out a certain functionality. Decomposition theory naturally provides the mathematical language to build an analytic foundation for the design of modularized and distributed control of networks.

1.2 Related Work

Decomposition in optimization is an old idea, and appears in early work on large-scale Linear Programing (LP) in the 1960s [3]. Ford and Fulkerson introduced some basic ideas into networks in 1962 [4] and Leon Lasdon used dual decomposition in his book to solve distributed problems in 1963 [5]. A good reference on decomposition methods is chapter 6 of Bertsekas [6].

The seminal publication on decomposition applied to networking problems was [7] by Kelly, Maulloo, and Tan in 1998. This paper outlines two major classes of approaches to solve the basic version of NUM: primal-based and dual-based. It is important to note that both approaches in [7] adopt a differential equation technique, analyzed through penalty functions and Lyapunov arguments, thus different from the language of primal and dual decomposition analyzed in chapter 3. After this paper, there was an explosion in literature about the NUM problem and its distributed algorithms.

Very good references are the work of Shakkottai and Srikant [8], the book of D.P. Palomar and Y.C Edlar in 2010 [9] and especially chapter 9 "Cooperative distributed multi-agent optimization" from authors Angelia Nedic and Asuman Ozdaglar, and most recently the book of R. Srikant and L. Ying [10]. The framework of NUM has found many applications in network resource allocation and Internet congestion control protocols. The basic concepts on applying these algorithms into cross-layer optimization can be found in [11].

The idea of decomposition comes up in the context of solving linear equations, but goes by other names such as block elimination, Schur complement methods, or (for special cases) matrix inversion lemma (see [13, App. C]). The core idea, i.e., using efficient methods to solve subproblems, and combining the results in such a way as to solve the larger problem, is the same, but the techniques are a bit different.

Additional related work of this thesis examples will be pointed out in each Section.

1.3 Thesis Outline

The thesis is organized as follows :

- Chapter 2 is an introduction to basic concepts of convex optimization, and also an introduction to the concept of subgradient and the subgradient methods. Finally, we present the dual ascent method.
- Chapter 3 presents the Primal and Dual decomposition methods with examples.
- Chapter 4 combines the Primal and Dual decomposition methods into more complex NUM applications.
- In Chapter 5, we test the convergence of Dual decomposition algorithm with message passing by minimizing the size of data transferred with quantization.
- Finally, Chapter 6 has a conclusion, and some suggestions for future work.

Chapter 2

Convex Optimization: Basics

2.1 Convex Optimization

In order to recognize convex optimization problems in engineering applications, one must first be familiar with the basic concepts of convexity and the commonly used convex optimization models. This section provides a concise review of these optimization concepts, based on the work of Zhi-Quan Luo and Wei Yu in [12]. In addition, key concepts, as the Karush-Kun-Tucker optimality conditions and Lagrangian Duality, are reviewed and stated explicitly for each of the convex optimization models.

2.1.1 Basic Optimization Concepts

Convex Sets A set $S \subset \mathbb{R}^n$ is said to be convex if for any two points x, $y \in S$, the line segment joining x and y also lies in S. Mathematically, this is expressed as:

$$\theta x + (1-\theta)y \in S, \ \ \forall \theta \in [0,1] \text{ and } x, y \in S.$$

Many well-known sets are convex, for example, the unit ball $S = \{x \mid \|x\| \leq 1\}$. However, the unit sphere $S = \{x \mid \|x\| = 1\}$ is not convex since the line segment joining any two distinct points is no longer on the unit sphere. In general, a convex set must be a solid body, containing no holes, and always curve outward. Other examples of convex sets include ellipsoids, hypercubes, polyhedral sets, and so on. In the real line $\mathbb{R}$, convex sets correspond to intervals (open or closed). A very important property of convex sets is the fact that the intersection of any number (possibly infinite) of convex set remains convex. For example, the set $S = \{x \mid \|x\| \leq 1, x \geq 0\}$ is the intersection of the unit ball with the non-negative orthant ($\mathbb{R}^n_+$), both of which are convex. Thus, their intersection S is also convex. The union of two convex sets is typically non-convex.

Convex functions A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be convex if, for any two points $x, y \in \mathbb{R}^n$ and $\forall \theta \in [0, 1]$,

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y).$$

Geometrically, this means that, when restricted on the line segment joining x and y, the line joining $(x, f(x))$ and $(y, f(y))$ is always above the graph of function f. There are many examples of convex functions, including the commonly seen univariate functions $|x|$, e^x, x^2, as well as multivariate functions $a^T x + b$, $\|Ax\|^2$, where A, a, and b are given matrix/vector/scalar. We say f is concave if $-f$ is convex. The entropy function $-\sum_i x_i \log x_i$ is a concave function over $\mathbb{R}^n$. If f is differentiable, then the convexity of f is equivalent to

$$f(y) \geq f(x) + \nabla f(x)^T (y - x),\ \forall x, y \in \mathbb{R}^n.$$

In other words, the first-order Taylor approximation serves as a global underestimator of f. Furthermore, if f is twice differentiable, then the convexity of f is equivalent to the positive semidefiniteness of its Hessian: $\nabla^2 f(x) \succeq 0,\ \forall x \in \mathbb{R}^n$. Thus, a linear function is always convex, while a quadratic function $x^T P x + a^T x + b$ is convex if and only if $P \succeq 0$. Notice that the linear plus the constant term $a^T x + b$ does not have any bearing on the convexity (or the lack of) of f. One can think of numerous examples of functions which are neither convex nor concave. For instance, the function x^3 is convex over $[0, \infty)$ and concave over the region $(-\infty, 0]$, but is neither convex nor concave over $\mathbb{R}$.

Important properties of convex functions are the facts that they are closed under summation, positive scaling, and pointwise maximum. In particular, if the $\{f_i\}$ are convex, then so is $\max_i\{f_i(x)\}$ (even though it is typically nondifferentiable). A notable connection between convex set and convex function is the fact that the level sets of any convex function f are always convex, i.e., $\{x \mid f(x) \leq c\}$ is convex for any $c \in \mathbb{R}$. The converse is not true, however. For example, the function $f(x) = \sqrt{|x|}$ is nonconvex, but its level sets are convex.

Convex Optimization Problems Consider a generic optimization problem (in the minimization form)

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0,\ i = 1, \ldots, m, \\ & h_j(x) = 0,\ j = 1, \ldots, r, \\ & x \in S \end{aligned} \tag{2.1}$$

where f_0 is called the objective function (or cost function), $\{f_i\}_{i=1}^m$ and $\{h_j\}_{j=1}^r$ are called the inequality and equality constraint functions, respectively, and S is called the constraint set. In practice, S can be implicitly defined by an oracle such as a user-supplied software. The optimization variable $x \in \mathbb{R}^n$ is said to be feasible if $x \in S$ and it satisfies all the inequality and equality constraints. A feasible solution x^* is said to be globally optimal if $f_0(x^*) \leq f_0(x)$ for all feasible x. In contrast, a feasible point $\vec{x}$ is said to be locally optimal if there exists some $\epsilon > 0$ such that $f_0(\vec{x}) \leq f_0(x)$ for all feasible x satisfying $\|x - \vec{x}\| \leq \epsilon$. The optimization problem (2.1) is said to be convex if

1. The functions f_i $(i = 0, 1, 2, \ldots, m)$ are convex;
2. The functions h_j $(i = 0, 1, 2, \ldots, r)$ are affine (i.e., h_j is of the form $a_j^T x + b_j$ for some $a_j \in \mathbb{R}^n$ and $b_j \in \mathbb{R}$); and
3. The set S is convex.

Violating any one of these three conditions results in a nonconvex problem. Notice that if we change "minimize" to "maximize" and change direction of the inequalities from "$f_i(x) \leq 0$" to "$f_i(x) \geq 0$" then (2.1) is convex if and only if all $f_i(x)$ $(i = 0, 1, 2, \ldots, m)$ are concave. For example, the following entropy maximization problem is convex:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \sum_{i=1}^n x_i \log x_i \\ \text{subject to} \quad & \sum_{i=1}^n x_i = 1,\ x_i \geq 0,\ i = 1, 2, \ldots, n, \\ & Ax = b, \end{aligned}$$

where the linear equalities $Ax = b$ represent the usual moment matching constraints.

Les us now put into perspective the role of convexity in optimization. It is well known that, for the problem of solving a system of equations, linearity is the dividing line between

the "easy" and "difficult" problems[1]. Once a problem is formulated as a solution to a system of a linear equations, the problem is considered solved since we can simply compute a solution analytically or using existing numerical software. In fact, there are many efficient and reliable software packages available for solving systems of linear equations, but none for nonlinear equations. The lack of high-quality software for solving nonlinear equations is merely a reflection of the fact that they are intrinsically difficult to solve.

In contrast, the dividing line between the "easy" and "difficult" problems in optimization is convexity. Convex optimization problems are the largest subclass of optimization problems which are efficiently solvable, whereas nonconvex optimization problems are generally difficult. The theory, algorithms, and software tools for convex optimization problems have advanced significantly over the last 50 years. There are now (freely downloadable) high-quality software packages which can deliver accurate solutions efficiently and reliably without the usual headaches of initialization, step-size selection or the risk of getting trapped in a local minimum. Once an engineering problem is formulated as a convex optimization problem, it is reasonable to consider it "solved."

For any convex optimization problem, the set of global optimal solutions is always convex. Moreover, every local optimal solution is also a global optimal solution, so there is no danger of being stuck at a local solution. There are other benefits associated with a convex optimization formulation. First, there exist highly efficient interior-point optimization algorithms whose worst-case complexity (i.e., the total number of arithmetic operations required to find an ϵ-optimal solution) grows gracefully as a polynomial function of the problem dimension. In addition, there exists an extensive duality theory for convex optimization problems, a consequence of which is the existence of a computable mathematical certificate for infeasible convex optimization problems. As a result, well-designed software for solving convex optimization problems typically return either an optimal solution or a certificate (in the form of a dual vector) that establishes the infeasibility of the problem. The latter property is extremely valuable in engineering design since it enables us to identify constraints which are too restrictive.

[1]This notions can be made precise using the computational complexity theory.

2.1.2 Lagragian Duality and Karush-Kuhn-Tucker Conditions

Consider the following (not necessarily convex) optimization problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0,\ i = 1, 2, \ldots, m, \\ & h_j(x) = 0,\ j = 1, 2, \ldots, r, \\ & x \in S. \end{aligned} \tag{2.2}$$

Let p^* denote the global minimum value of (2.2). For symmetry reasons, we will call (2.2) the primal optimization problem, and call x the primal vector. Introducing dual variables $\lambda \in \mathbb{R}^m$ and $\nu \in \mathbb{R}^r$, we can form the Lagrangian function

$$L(x, \lambda, \nu) := f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{r} \nu_j h_j(x).$$

The so-called dual function $g(\lambda, \nu)$ associated with (2.2) is defined as

$$g(\lambda, \nu) := \min_{x \in S} L(x, \lambda, \nu).$$

Notice that, as a pointwise minimum of a family of linear functions (in (λ, ν)), the dual function $g(\lambda, \nu)$ is always concave. We will say (λ, ν) is dual feasible if $\lambda \geq 0$ and $g(\lambda, \nu)$ is finite. The well-known weak duality result says the following.

Proposition 1 For any primal feasible vector x and any dual feasible vector (λ, ν), it holds true that

$$f_0(x) \geq g(\lambda, \nu).$$

In other words, for any dual feasible vector (λ, ν), the dual function value $g(\lambda, \nu)$ always serves as a lower bound on the primal objective value $f_0(x)$. Notice that x and (λ, ν) are chosen independent from each other (so long as they are both feasible). Thus, $p^* \geq g(\lambda, \nu)$ for all dual feasible vector (λ, ν). The largest lower bound for p^* can be found by solving the following dual optimization problem:

$$\begin{aligned} \underset{\lambda, \nu}{\text{maximize}} \quad & g(\lambda, \nu) \\ \text{subject to} \quad & \lambda \geq 0,\ \nu \in \mathbb{R}. \end{aligned} \tag{2.3}$$

Notice that the dual problem (2.3) is always convex regardless of the convexity of the primal problem (2.2), since $g(\lambda, \nu)$ is concave. Let us denote the maximum value of (2.3) by d^*. Then, we have $p^* \geq d^*$. Interestingly, for most convex optimization problems (satisfying some mild constraint qualification conditions, such as the existence of a strict interior feasible point), we actually have $p^* = d^*$. This is called strong duality.

In general, the dual function $g(\lambda, \nu)$ is difficult to compute. However, for special classes of convex optimization problems, we can derive their duals explicitly.

Next, we present a local optimality condition for the optimization problem (2.2). For ease of exposition, let us assume $S = \mathbb{R}^n$. Then, a necessary condition for x^* to be a local optimal solution of (2.2) is that there exists some (λ^*, ν^*) such that

$$f_i(x^*) \leq 0,\ i = 1, 2, \ldots, m \tag{2.4}$$

$$h_j(x^*) = 0,\ j = 1, 2, \ldots, r \tag{2.5}$$

$$\lambda^* \geq 0, \tag{2.6}$$

$$\lambda_i^* f_i(x^*) = 0, i = 1, 2, \ldots, m \tag{2.7}$$

and

$$\nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{j=1}^{r} \nu_j^* \nabla h_j(x^*) = 0. \tag{2.8}$$

Collectively, the conditions (2.4)-(2.8) are called the Karush-Kuhn-Tucker (KKT) optimality conditions. Notice that conditions (2.4) and (2.5) guarantee primal feasibility of x^*, condition (2.6) guarantees dual feasibility, condition (2.7) signifies the complementary slackness for the primal and dual inequality constraint pairs, $f_i(x) \leq 0$ and $\lambda_i \geq 0$, while condition (2.8) is equivalent to $\nabla_x L(x^*, \lambda^*, \nu^*) = 0$.

In general, the KKT conditions are necessary but not sufficient for optimality. However, for convex optimization problems (and under mild constraint qualification conditions), the KKT conditions are also sufficient. If the constraints in (2.2) are absent, the corresponding KKT condition simply reduces to the well-known stationarity condition for unconstrained optimization problems: $\nabla f_0(x^*) = 0$. That is, an unconstrained local minimum must be attained at a stationary point (at which the gradient of f_0 vanishes). However, in the presence of constraints, local optimal solution of (2.2) is no longer attained at a stationary point; instead, it is attained at a KKT point x^*, which, together with some dual feasible vectors, satisfies the KKT conditions (2.4)$-$(2.8).

Detecting infeasibility Efficient detection of infeasibility is essential in engineering design applications. However, the problem of detecting and removing the incompatible constraints is NP-hard in general, especially if the constraints are nonconvex. However, for convex constraints, we can make use of duality theory to prove inconsistency. Let us consider the following example.

Example 2.1. Determine if the following linear system is feasible:

$$\begin{aligned} x_1 + x_2 &\leq 1 \\ x_1 + x_2 &\leq -1 \\ -x_1 &\leq -1. \end{aligned}$$

Let us multiply the last inequality by 2 and add it to the first and the second inequalities. The resulting inequality is $0 \leq -1$, which is a contradiction. This shows that the above linear system is infeasible. ■

Modern software packages (e.g., SeDuMi) for solving convex optimization problems either compute an optimal solution or provide a certificate showing infeasibility. In contrast, software for nonconvex optimization problems cannot detect infeasibility. It typically fails to converge when the underlying problem is infeasible, either due to data overflow or because the maximum number of iterations is exceeded.

2.1.3 Linear Programming (LP)

We now review a commonly used convex optimization model in engineering design applications. Consider a primal-dual pair of optimization problems

$$\begin{aligned} &\text{minimize} && c^T x \\ &\text{subject to} && Ax = b,\ x \geq 0. \end{aligned} \tag{2.9}$$

Its dual is

$$\begin{aligned} &\text{minimize} && b^T y \\ &\text{subject to} && A^* y + s = c,\ s \geq 0. \end{aligned} \tag{2.10}$$

The optimality conditions are given by

$$Ax = b,\ x \geq 0,\ A^T y = s + c, s > 0, x^T s = 0.$$

2.2 Subgradient

In this section, based on Stephen P. Boyd's notes [14], we review the concept of the subgradient and present the subgradient method, one of the basic algorithms used for non-differentiable problems, which makes the subgradient method popular on decomposition methods.

2.2.1 Definition

We say a vector $g \in \mathbb{R}^n$ is a *subgradient* of $f : \mathbb{R}^n \to \mathbb{R}$ at $x \in \mathbf{dom} f$ if, for all $z \in \mathbf{dom} f$,

$$f(z) \geq f(x) + g^T(z - x). \tag{2.11}$$

If f is convex and differentiable, then its gradient at x is a subgradient. But a subgradient can exist even when f is not differentiable at x, as illustrated in Figure 2.1. The same example shows that there can be more than one subgradients of a function f at a point x.

There are several ways to interpret a subgradient. A vector g is a subgradient of f at x if the affine function (of z) $f(x) + g^T(z - x)$ is a global underestimator of f. Geometrically, g is a subgradient of f at x if $(g, -1)$ supports $\mathbf{epi} f$ at $(x, f(x))$.

A function f is called *subdifferentiable* at x if there exists at least one subgradient at x. The set of subgradients of f at the point x is called the *subdifferential* of f at x, and is denoted $\partial f(x)$. A function f is called subdifferentiable if it is subdifferentiable at all $x \in \mathbf{dom} f$.

Example 2.2. Absolute value. Consider the function $f(x) = |x|$. For $x < 0$, the subgradient is unique: $\partial f(x) = \{-1\}$. Similarly, for $x > 0$, we have $\partial f(x) = \{1\}$. At $x = 0$, the subdifferential is defined by the inequality $|x| \geq gx$ for all x, which is satisfied if and only if $g \in [-1, 1]$. Therefore we have $\partial f(0) = [-1, 1]$. This is illustrated in Figure 2.2. ■

2.2.2 Basic properties

- The subdifferential $\partial f(x)$ is always a closed convex set, even if f is not convex. This follows from the fact that it is the intersection of an infinite set of halfspaces:

$$\partial f(x) = \bigcap_{z \in \mathbf{dom} f} \{g \mid f(z) \geq f(x) + g^T(z - x)\}.$$

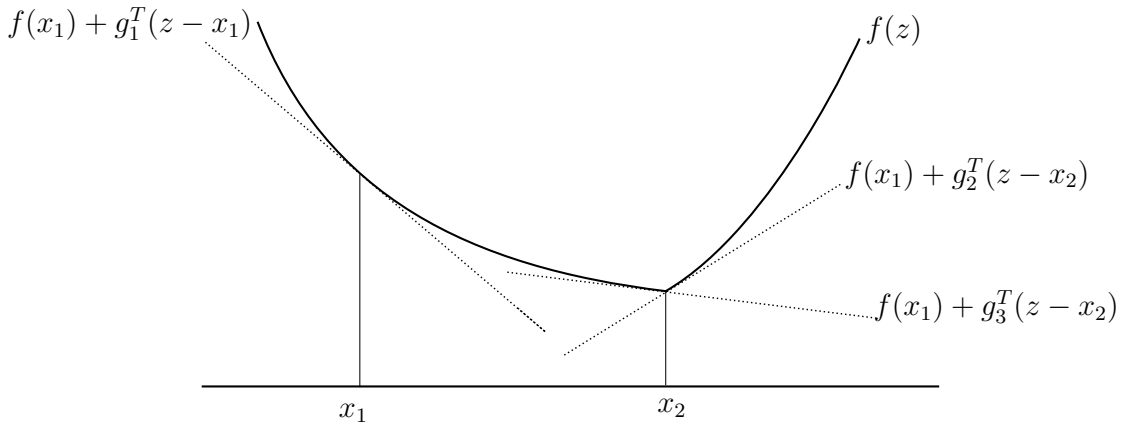

Figure 2.1: At x_1, the convex function f is differentiable, and g_1 (which is the derivative of f at x_1) is the unique subgradient at x_1. At x_2, f is not differentiable. At this point, f has many subgradients: two subgradients, g_2 and g_3, are shown.

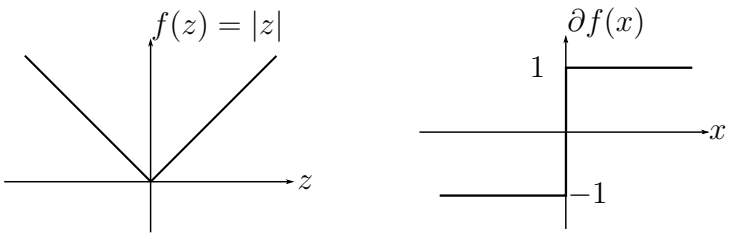

Figure 2.2: The absolute value function (left) and its subdifferential $\partial f(x)$ as a multivalued function of x (right).

- If f is convex and differentiable, then $\nabla f(x)$ is a subgradient of f at x.
- If $f(y) \leq f(x) + g^T(y - x)$ for all y, then g is a **supergradient**.

2.2.3 Calculus of subgradients

In this section, we describe rules for constructing subgradients of convex functions. We will distinguish two levels of detail. In the "weak" calculus of subgradients, the goal is to produce one subgradient, even if more subgradients exist. This is sufficient in practice, since subgradient, localization, and cutting-plane methods require only one subgradient at any point. A second and much more difficult task is to describe the complete set of subgradients $\partial f(x)$ as a function of x. We call this the "strong" calculus of subgradients. It is useful in theoretical investigations, for example, when describing the precise optimality conditions.

Nonnegative scaling For $\alpha \geq 0$, $\partial(\alpha f)(x) = \alpha \partial f(x)$.

Sum and integral Suppose $f = f_1 + f_2 + \cdots + f_m$, where $f_1 + f_2 + \cdots + f_m$ are convex functions. Then, we have

$$\partial f(x) = \partial f_1(x) + \partial f_2(x) + \cdots + \partial f_m(x).$$

This property extends to infinite sums, integrals, and expectations (provided they exist).

Affine transformations of domain Suppose f is convex, and let $h(x) = f(Ax + b)$. Then $\partial h(x) = A^T \partial f(Ax + b)$.

2.3 Subgradient method

2.3.1 Introduction

The subgradient method is a very simple algorithm for the minimization of a nondifferentiable convex function. The method looks very much like the ordinary gradient method for differentiable functions, but with several notable exceptions:

- The subgradient method applies directly to nondifferentiable f.
- The step lengths are not chosen via line search, as in the ordinary gradient method. In the most common cases, the step lengths are fixed ahead of time.
- Unlike the ordinary gradient method, the subgradient method is not a descent method; the function value can (and often does) increase.

The subgradient method is readily extended to handle problems with constraints.

2.3.2 Basic subgradient method

The goal is to minimize a nondifferentiable convex function $f : \mathbb{R}^n \to \mathbb{R}$. To do this, the subgradient method uses the simple iteration

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)},$$

where

- $x^{(k)}$ is the kth iterate,
- $g^{(k)}$ is **any** subgradient of f at $x^{(k)}$,
- $\alpha_k > 0$ is the k_{th} stepsize.

It may happen that $-g^{(k)}$ is not a descent direction for f at $x^{(k)}$. In such cases, we always have $f(x^{(k+1)}) > f(x^{(k)})$. Even when $-g^{(k)}$ is a descent direction at $x^{(k)}$, the step size can be such that $f(x^{(k+1)}) > f(x^{(k)})$. In other words, one iteration of the subgradient method may increase the objective function. Since the subgradient method is not a descent method, at each time instant, it is common to keep track of the best point until this time instant.

2.3.3 Step size rules

In the subgradient method, the step size selection is very different from the standard gradient method. Many different types of step size rules are used. We will start with five basic step size rules.

- Constant step size: $\alpha_k = \alpha$ is a positive constant, independent of k.
- Constant step length: $\alpha_k = \gamma/\|g^{(k)}\|$, where $\gamma > 0$. This means that $\|x^{(k+1)} - x^{(k)}\|_2 = \gamma$.
- Square summable but not summable: The step sizes satisfy

$$\sum_{k=1}^{\infty} \alpha_k^2 < \infty, \quad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

  One typical example is $\alpha_k = \alpha/(b+k)$, where $a > 0$ and $b \geq 0$.

- Nonsummable diminishing: The step sizes satisfy

$$\lim_{x \to \infty} \alpha_k = 0, \quad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

  Step sizes that satisfy this condition are called diminishing step size rules. A typical example is $\alpha_k = \alpha/\sqrt{k}$, where $a > 0$.

- Nonsummable diminishing step lengths: The step sizes are chosen as $\alpha = \gamma_k / \|g^{(k)}\|_2$, where

$$\gamma_k \geq 0, \quad \lim_{k\to\infty} \gamma_k = 0, \quad \sum_{k=1}^{\infty} \gamma_k = \infty.$$

There are still other choices, and many variations of these choices. For example, when f^* is known, we can use Polyak's step size

$$\alpha_k = \frac{f(x^{(k)}) - f^*}{\|g^{(k)}\|_2^2}.$$

The most interesting feature of these choices is that they are determined before the algorithm is run; they do not depend on any data computed during the algorithm. This is very different from the step size rules found in standard descent methods, which very much depend on the current point and search direction.

2.3.4 Convergence

Under some technical conditions (boundedness of optimum value, boundedness of subgradients by G), the limiting value of the subgradient method $\hat{f} = \lim_{k\to\infty} f_{\text{best}}^{(k)}$ satisfies:

- constant stepsize: $\hat{f} - f^* \leq G^2\alpha/2$ (suboptimal)
- constant step length: $\hat{f} - f^* \leq G\gamma/2$ (suboptimal)
- diminishing stepsize rule: $\hat{f} = f^*$ (converges)

2.3.5 Projected subgradient method

One extension of the subgradient method is the projected subgradient method, which solves the constrained convex optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{C} \end{array} \tag{2.12}$$

where $\mathcal{C}$ is a convex set. The projected subgradient method is given by

$$x^{(k+1)} = [x^{(k)} - \alpha_k g^{(k)}]_{\mathcal{C}}$$

where $[.]_{\mathcal{C}}$ is the (Euclidean) projection on $\mathcal{C}$, and $g^{(k)}$ is any subgradient of f at $x^{(k)}$. The step size rules described before can be used in this case as well, with similar convergence results. Note that $x^{(k)} \in \mathcal{C}$ i.e., $x^{(k)}$ is feasible.

2.4 Dual Ascent

In this section, based on [15], we briefly review the dual ascent algorithm which is a precursor of the Dual decomposition algorithm, which we will take a glimpse here, but will get a better look in the next chapter.

Consider the equality-constrained convex optimization problem

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && Ax = b, \end{aligned} \tag{2.13}$$

with variable $x \in \mathbb{R}^n$, where $A \in \mathbb{R}^{m \times n}$ and $f : \mathbb{R}^n \to \mathbb{R}$ is convex. The Lagrangian for problem (2.13) is

$$L(x, y) = f(x) + y^T(Ax - b)$$

and the dual function is

$$g(y) = \inf_x L(x, y)$$

where y is the dual variable or Lagrange multiplier. The dual problem is

$$\text{maximize} \quad g(y) \tag{2.14}$$

with variable $y \in \mathbb{R}^m$. Assuming that strong duality holds, the optimal values of the primal and dual problems are equal. We can recover a primal optimal point x^* from a dual optimal point y^* as

$$x^* = \arg\min_x L(x, y^*),$$

provided there is only one minimizer of $L(x, y^*)$. (This is the case if, *e.g.,* f is strictly convex.)

The dual ascent method is an iterative technique for the solution of (2.13) that can be described as follows. Assuming that (1) dual function g is differentiable and (2) at the k-th

iteration, we have computed dual variable y^k, the $(k+1)$-st iteration is given by

$$x^{k+1} := \arg\min_x L(x, y^k) \tag{2.15}$$

$$y^{k+1} := y^k + \alpha^k \nabla g(y^k), \tag{2.16}$$

where $\alpha^k > 0$ is a step size. The first step (2.15) is an x-minimization step, and the second step (2.16) is a dual variable update. In our case, $g(y^k) = f(x^{k+1}) + (y^k)^T(Ax^{k+1} - b)$ and $\nabla g(y^k) = Ax^{k+1} - b$.

The dual variable y can be interpreted as a vector of prices, and the y-update is then called a *price update* or *price adjustment* step. This algorithm is called dual ascent since, with appropriate choice of α^k, the dual function increases in each step, *i.e.*, $g(y^{k+1}) > g(y^k)$.

The dual ascent method can be used even in some cases when g is not differentiable. In these cases, the residual $Ax^{k+1} - b$ is not the gradient but the negative of a subgradient of $-g(y^k)$. These cases require a different choice of α^k than when g is differentiable, and convergence is not monotone; it is often the case that $g(y^{k+1}) \not> g(y^k)$. In these cases, the algorithm is called the dual subgradient method (see previous section). If α^k is chosen appropriately and several other assumptions hold, then x^k converges to an optimal point and y^k converges to an optimal dual point. However, these assumptions do not hold in many applications, so dual ascent often cannot be used.

The major benefit of the dual ascent method is that it can lead to a decentralized algorithm in some cases. Suppose, for example, that the objective f is separable (with respect to a partition or splitting of the variable into subvectors), meaning that

$$f(x) = \sum_{i=1}^{N} f_i(x_i),$$

where $x = (x_1, \ldots, x_N)$ and the variables $x_i \in \mathbb{R}^{n_i}$ are subvectors of x. Partitioning matrix A conformably as

$$A = [A_1, \ldots, A_N],$$

so $Ax = \sum_{i=1}^{N} A_i x_i$, the Lagrangian can be written as

$$L(x, y) = \sum_{i=1}^{N} L_i(x_i, y) = \sum_{i=1}^{N} (f_i(x_i) + y^T A_i x_i - (1/N) y^T b)$$

which is also separable in x. This means that the x-minimization step (2.15) splits into N

separate problems that can be solved in parallel. Explicitly, the algorithm is

$$x_i^{k+1} := \arg\min_{x_i} L_i(x_i, y^k),\ i = 1, \ldots, N \tag{2.17}$$

$$y^{k+1} := y^k + \alpha^k (Ax^{k+1} - b), \qquad . \tag{2.18}$$

The x-maximization step (2.17) is carried out independently, in parallel, for each $i = 1, \ldots, N$. In this case, we refer to the dual ascent method as Dual Decomposition.

Chapter 3

Decomposition Methods

3.1 Introduction

The idea is to decompose the original large problem into subproblems (locally solved) and a master problem (coordinating the subproblems) where there is communication between the master problem and the subproblems (Figure 3.1).

The two main classes of decomposition techniques are the **primal decomposition** and the **dual decomposition**.

In primal decomposition, we decompose the original problem by optimizing over one set of variables and then over the remaining set, where the master problem directly allocates the existing resources to subproblems.

In dual decomposition, we decompose the dual problem obtained after a Lagrange relaxation of the coupling constrains, where the master problem sets prices for the resources to the subproblems.

We use decomposition methods for various reasons:

- to allow the solution of a problem otherwise unsolvable for memory reasons (useful in areas such as biology or image processing),
- to speed up the solution of the problem via parallel computation,
- to solve the problem in a distributed way (desirable for some wireless networks),
- to derive nice, insightful, and efficient numerical algorithms as alternatives to general-purpose, e.g., interior-point, methods.

Problems for which decomposition works in one step are called (block) separable, or trivially parallelizable. We can see this in the following example that decouples naturally.

Example 3.1. Consider the following problem:

$$\begin{array}{ll} \underset{x_1,x_2}{\text{minimize}} & f_1(x_1) + f_2(x_2) \\ \text{subject to} & x_1 \in \mathcal{X}_1, \;\; x_2 \in \mathcal{X}_2. \end{array} \tag{3.1}$$

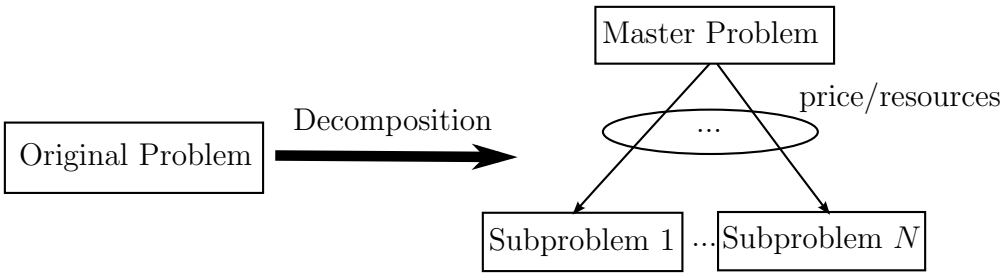

Figure 3.1: Decomposition idea.

It is obvious that we can solve each problem separately (and in parallel), and then re-assemble the solution x. ■

Of course, this is a trivial, and not too interesting, case. A more interesting situation occurs when there is some coupling or interaction between the subvectors, so the problems cannot be solved independently. For these cases, there are techniques that solve the overall problem by iteratively solving a sequence of smaller problems.

3.2 Primal Decomposition

A primal decomposition is appropriate when the problem has a coupling variable such that, when fixed to some value, the rest of the optimization problem decouples into several subproblems.

Consider the following problem

$$\begin{array}{ll} \underset{x_1,x_2,y}{\text{minimize}} & f_1(x_1,y)+f_2(x_2,y) \\ \text{subject to} & x_1\in\mathcal{X}_1,\ \ x_2\in\mathcal{X}_2, \\ & y\in\mathcal{Y}, \end{array} \tag{3.2}$$

where y is the *complicating* or *coupling* variable, and x_1 and x_2 are *local* or *private* variables. When y is fixed, the problem is separable in x_1 and x_2 and then decouples into two subproblems that can be solved independently.

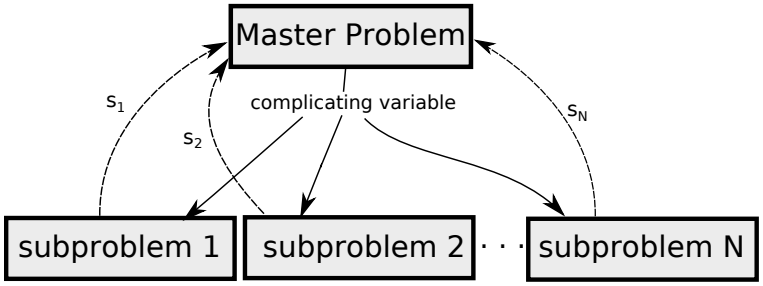

Figure 3.2: The interaction between the Master Problem and the subproblems.

For a given y, we define the subproblems

$$\text{subproblem 1}: \quad \underset{x_1 \in \mathcal{X}_1}{\text{minimize}}\ f_1(x_1, y)$$

$$\text{subproblem 2}: \quad \underset{x_2 \in \mathcal{X}_2}{\text{minimize}}\ f_2(x_2, y)$$

with optimal values $f_1^*(y)$ and $f_2^*(y)$.

Since $\min_{x,y} f \equiv \min_y \min_x f$ it follows that the original problem is equivalent to the master primal problem:

$$\underset{y \in \mathcal{Y}}{\text{minimize}}\ f_1^*(y) + f_2^*(y)$$

If the original problem is convex, so is the master problem. In general, there are many methods to solve the master problem such as bisection, subgradient, cutting-plane, ellipsoid. The subgradient method is commonly used because it is very simple and allows itself to distributed implementation; however, its convergence is slow in practice. For constrained problems, we use the projected subgradient method (2.3.5). In Figure 3.2, we can see the communication between the master problem and the subproblems.

Algorithm 3.1 Primal Decomposition Algorithm

1: Solve subproblems (possibly in parallel)
2: Calculate subgradients
3: Update the complicating variable that reduces the price of the primal master problem
4: Repeat until convergence

3.2.1 Primal Decomposition Application

We consider the Linear Programming example from [16], with variables u and v,

$$\begin{array}{ll} \text{minimize} & f(u,v) = c_1^T u + c_2^T v \\ \text{subject to} & A_1 u \preceq b_1 \\ & A_2 v \preceq b_2 \\ & F_1 u + F_2 v \preceq h. \end{array} \tag{3.3}$$

We observe that $F_1u+F_2v \preceq h$ is the coupling constraint; removing it allows problem (3.3) to be solved via two separate LPs.

In order to solve problem (3.3), we introduce variable z and express $F_1u + F_2v \preceq h$ as

$$F_1 u \preceq z, \quad F_2 v \preceq h - z.$$

Variable z sets the allocation of resources between the two subproblems and the original problem is equivalent to the **master problem**

$$\text{minimize}_z \ \phi_1(z) + \phi_2(z) \tag{3.4}$$

where

$$\begin{aligned} \phi_1(z) &= \inf_u \{c_1^T u | A_1 u \preceq b_1, F_1 u \preceq z\} \\ \phi_2(z) &= \inf_v \{c_2^T v | A_2 v \preceq b_2, F_2 v \preceq h - z\}. \end{aligned} \tag{3.5}$$

We solve problem (3.4) iteratively. We observe that, for a fixed $z^{(t)}$, the two LPs in (3.5) are independent and thus can be solved separately. After their solution, we update $z^{(t)}$ using a subgradient step as

$$z^{(t+1)} = z^{(t)} - \alpha^{(t)} g^{(t)} \tag{3.6}$$

where $g^{(t)}$ is a subgradient of $\phi_1(z) + \phi_2(z)$. This subgradient can be computed as follows.

Consider the dual problem of $\phi_1(z)$ by relaxing $F_1u \preceq z$,

$$\begin{array}{ll} \underset{u,\lambda \succeq 0}{\text{minimize}} & c_1^T u + \lambda_1^T (F_1 u - z) \\ \text{subject to} & A_1 u \preceq b_1. \end{array} \tag{3.7}$$

Consider $p^*(z)$ is the optimal value of problem (3.7), with optimal (u^*, λ_1^*), us a function of z. So, the function is $p^*(z) = c_1^T u^* + \lambda_1^{*T}(F_1 u^* - z)$. The gradient is given by $\nabla_z p(z) = -\lambda_1^*$. This gradient shows us in which direction to move z in order to minimize the dual function. If we have strong duality (which we have in this case), the above gradient is a descent direction for function $\phi_1(z)$ as well. Likewise, we can prove that for $\phi_2(z)$ the subgradient is λ_2 and since the subgradient of the summation of two functions is the summation of their subgradients (from section 2.2.3), this concludes the proof that the dual variables λ_1 and λ_2 of the constraints $F_1 u \preceq z$ and $F_2 v \preceq h - z$ can be used for a subgradient given by

$$g = -\lambda_1 + \lambda_2 \in \partial(\phi_1(z) + \phi_2(z)). \tag{3.8}$$

Summarizing, we solve problem (3.3) iteratively using Primal Decomposition. In the t-th iteration we update

$$u^{(t+1)} := \arg\min_u \{c_1^T u | A_1 u \preceq b_1, F_1 u \preceq z^{(t)}\} \tag{3.9}$$

$$v^{(t+1)} := \arg\min_v \{c_2^T v | A_2 v \preceq b_1, F_1 v \preceq z^{(t)}\} \tag{3.10}$$

$$z^{((t+1))} := z^{(t)} - \alpha^{(t)} g^{(t)} \tag{3.11}$$

3.2.2 Numerical Example

We solved problem (3.3) for $n_u = n_v = 10$ variables, $m_u = m_v = 100$ private inequalities, and $p = 5$ complicating inequalities, by using the decomposition approach we described above. In the subgradient method, we used (1) a diminishing step size $a_t = 0.01/\sqrt{t}$ and (2) a constant step size $a = 0.001$.

In the left plot of Figure 3.3, we show with red line the difference of the cost function at iteration t, $f^{(t)} = \phi_1(z^{(t)}) + \phi_2(z^{(t)})$, from its optimal value f^*, versus the iteration number t, for step size $a_t = 0.01/\sqrt{t}$, while with blue line we show the difference from f^* of the best value of $f^{(t)}$ for time up to t. In the right plot of Figure 3.3, we show $f^{(t)}$ versus t. In Figure 3.4, we show the corresponding quantities for a constant step size $a = 0.001$.

As we can see, the algorithm converges to the optimal price, and we can achieve error less than 10^{-3} in less than 200 iterations.

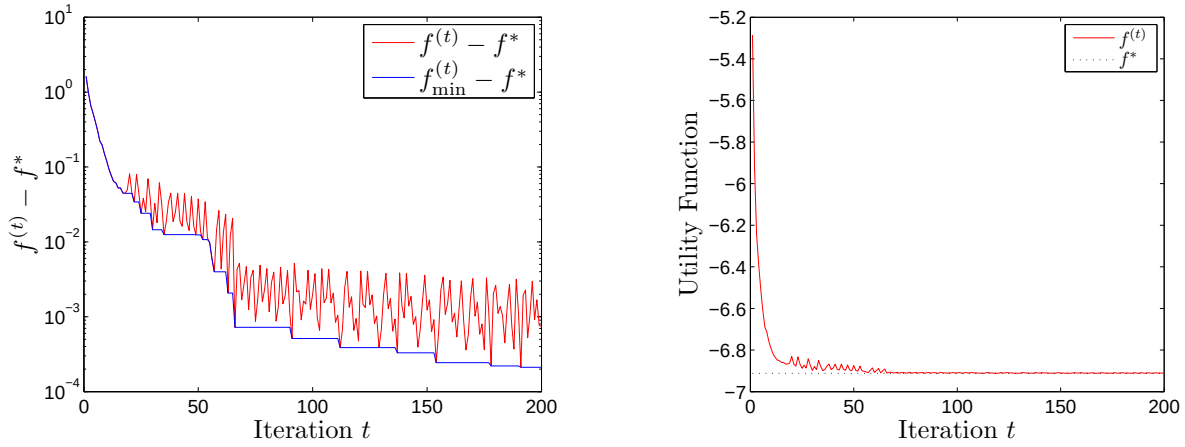

Figure 3.3: Left. With red line we plot quantity $f^{(t)} - f^*$, while with blue line we plot $\min_{k<t} f^{(k)} - f^*$, versus iteration number t, for $\alpha_t = 0.01\sqrt{t}$. Right: We plot the utility function $f^{(t)}$ versus iteration number t.

3.3 Dual Decomposition

3.3.1 The Basic NUM

A dual decomposition is appropriate when the problem has a coupling constraint such that, when relaxed, the optimization problem decouples into several subproblems. We illustrate the full dual decomposition by applying the standard method to the basic Network Utility Maximization (NUM) for a distributed end-to-end rate allocation.

The problem formulation is as follows. There exist S sources, with fixed routes, in a network with L links. The l-th link, for $l = 1, \ldots, L$, has a fixed capacity c_l. Source s, for $s = 1, \ldots, S$, (see Figure 3.5)

1. transmits at a rate $x_s \geq 0$;
2. emits one flow using a fixed set of links in its path denoted as $L(s)$;
3. has utility function $U_s(x_s)$ which is twice-differentiable, increasing, and strictly concave function.

We have to maximize the total utility $f(x) = \sum_s U_s(x_s)$, subject to linear flow con-

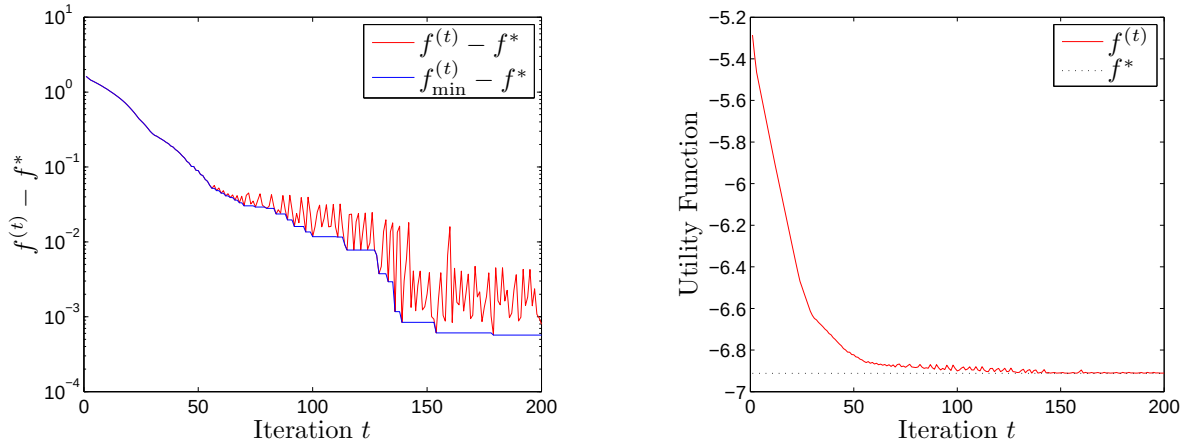

Figure 3.4: Left. With red line we plot quantity $f^{(t)} - f^*$, while with blue line we plot $\min_{k<t} f^{(k)} - f^*$, versus iteration number t, for $\alpha = 0.001$. Right: We plot the utility function $f^{(t)}$ versus iteration number t.

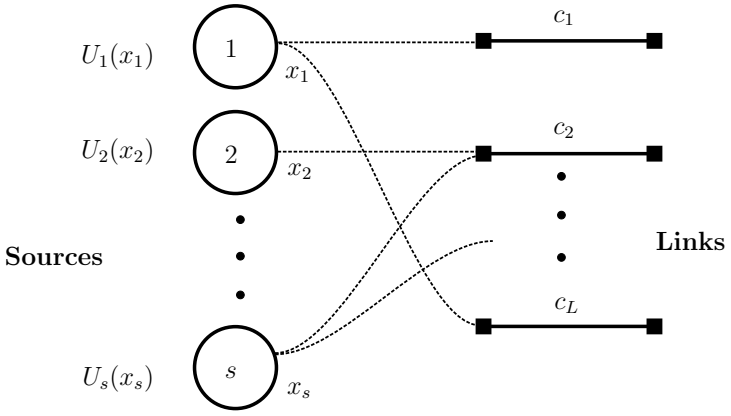

Figure 3.5: Network formulation. Each source is connected with all links in its path.

strains $\sum_{s:l\in L(s)} x_s \leq c_l$, for all links l, that is

$$\begin{aligned} \underset{x\succeq 0}{\text{maximize}} \quad & \sum_s U_s(x_s) \\ \text{subject to} \quad & \sum_{s:l\in L(s)} x_s \leq c_l, \ \forall l. \end{aligned} \tag{3.12}$$

In order to simplify the formulation, we define a *flow-link incidence* matrix $A \in \mathbb{R}^{l\times s}$ where

$$A_{i,j} = \begin{cases} 1, & \text{if } j_{th} \text{ source flow passes through } i_{th} \text{ link} \\ 0, & \text{otherwise.} \end{cases}$$

Hence

$$\sum_{s:l\in L(s)} x_s \leq c_l \equiv Ax \preceq c.$$

So, we can express (3.12) as

$$\begin{array}{ll} \underset{x\succeq 0}{\text{maximize}} & \sum_s U_s(x_s) \\ \text{subject to} & Ax \preceq c. \end{array} \tag{3.13}$$

We can solve (3.12) in a **centralized** manner with the Dual Ascent method of section 2.4. We first form the Lagrangian

$$L(x,\lambda) = \sum_s U_s(x_s) + \sum_l \lambda_l \left(c_l - \sum_{s:l\in L(s)} x_s \right) \tag{3.14}$$

The Dual Ascent algorithm 3.2 for solving this problem converges under strong duality.

Algorithm 3.2 Dual Ascent Algorithm

1: Choose a λ
2: Solve $x = \arg\min L(x,\lambda)$
3: Update the Lagrange multiplier $\lambda = \lambda - \alpha\nabla g(\lambda)$
4: Repeat from 2 until convergence

It can be shown that the update of λ in step 3 is given by

$$\lambda_l(t+1) = \left[\lambda_l(t) - \alpha\left(c_l - a_l^T x\right)\right]^+,\ l = 1,\ldots,L. \tag{3.15}$$

The proof is based on the following lemma.

Lemma 3.1. *Let $g(\lambda)$ be the dual function corresponding to the problem*

$$\begin{array}{ll} \underset{x}{\textit{minimize}} & f_0(x) \\ \textit{subject to} & h_i(x) \leq 0, \quad i = 0,\ldots,m. \end{array}$$

A subgradient of $g(\lambda)$ is $h(x^*(\lambda))$, where $x^*(\lambda)$ is the optimal x for a fixed λ.

Proof. The Lagrangian is $L(x,\lambda) = f_0(x) + \lambda^T h(x)$ and the dual function $g(\lambda) = \inf_x L(x,\lambda)$.

Then,

$$\begin{aligned} g(\lambda) &= \inf_x (f_0(x) + \lambda^T h(x)) \\ &\leq f_0(x^*(\lambda_0)) + \lambda^T h(x^*(\lambda_0)) \\ &= f_0(x^*(\lambda_0)) + \lambda_0^T h(x^*(\lambda_0)) + (\lambda - \lambda_0)^T h(x^*(\lambda_0)) \\ &= g(\lambda_0) + (\lambda - \lambda_0)^T h(x^*(\lambda_0)). \end{aligned}$$

Thus, $h(x^*(\lambda_0))$ is a subgradient of g at λ_0. □

If we want to solve (3.12) in a **distributive** manner, we use Dual Decomposition. At first, we express the Lagrangian (3.14) as

$$\begin{aligned} L(x,\lambda) &= \sum_s U_s(x_s) + \sum_l \lambda_l \left(c_l - \sum_{s:l\in L(s)} x_s \right) \\ &= \sum_s U_s(x_s) + \sum_l \lambda_l \left(c_l - a_l^T x \right) \\ &= \sum_s U_s(x_s) - \sum_l \lambda_l (a_l^T x) + \sum_l \lambda_l c_l \\ &= \sum_s U_s(x_s) - \sum_s x_s \sum_l \lambda_l a_{l,s} + \sum_l \lambda_l c_l \\ &= \sum_s \left(U_s(x_s) - \lambda_{(s)} x_s \right) + \sum_l \lambda_l c_l \\ &= \sum_s L_s(x_s, \lambda_{(s)}) + \sum_l c_l \lambda_l \end{aligned} \tag{3.16}$$

where a_l^T is the l_{th} row of matrix A, λ_l is the Lagrange multiplier (link price) associated with the linear flow constraint on link l,

$$\lambda_{(s)} = \sum_{l\in L(s)} \lambda_l$$

is the aggregate path congestion price of those links used by source s, and

$$L_s(x_s, \lambda_{(s)}) = U_s(x_s) - \lambda_{(s)} x_s$$

is the s_{th} Lagrangian to be maximized for the s_{th} source. In (3.16) we observe that, for a fixed λ, the Lagrangian $L(x,\lambda)$ is separable with respect to x_s.

Thus, for source s, for $s = 1, \ldots, S$, we solve the problem

$$\underset{x_s}{\text{maximize}} \quad L_s(x_s, \lambda_{(s)}) = U_s(x_s) - \lambda_{(s)} x_s.$$

These problems can be solved in parallel. Then, using the optimal values of these problems, we update λ via the subgradient step (see (3.15)).

Summarizing, we solve problem (3.12) iteratively using Dual Decomposition. In the t-th iteration we update

$$x_s^{(t+1)} := \arg\max_{x_s} \{U_s(x_s) - \lambda_{(s)}^{(t)} x_s\}, \ s = 1, \ldots, S. \tag{3.17}$$

$$\lambda_l^{(t+1)} := \left[\lambda_l^{(t)} - \alpha \left(c_l - a_l^T x^{(t+1)}\right)\right]^+, \ l = 1, \ldots, L. \tag{3.18}$$

3.3.2 Numerical Results

The utility function for our example is $U_s(x_s) = cost_s \sqrt{x_s}$, with random $cost_s \in \mathbb{R}^+$. We decompose the dual problem (3.13) to a master problem and S subproblems by using the Lagrange multiplier as a link price between the problems. For each source, we solve the corresponding subproblem for a fixed λ starting with $\lambda = 1_l$[1]

$$\underset{x_s}{\text{maximize}} \quad cost_s \sqrt{x_s} - \lambda_s^{(t)} x_s, \quad s = 1, \ldots, S, \tag{3.19}$$

where $\lambda_s^{(t)}$ is the aggregate Lagrange multiplier for all links that x_s passes through after t iterations and it can be easily calculated by $((\lambda^{(t)})^T A)_s$.

We experimented with step sizes α according to step size rules from Chapter 2. We used a fixed incidence matrix A with $links = 10$ and $sources = 5$.

In the left plot of Figure 3.6, we show with red line the difference of the utility function at iteration t, $f(t) = \sum_s U_s(x_s^{(t)})$, from its optimal value f^*, versus the iteration number t, for step size $a_t = 1/\sqrt{t}$, while with blue line we show the difference from the optimal resource allocation x^* and the resource allocation $x^{(t)}$ for the iteration t. In the right plot of Figure 3.6, we show $f(t)$ versus iteration t. In Figure 3.7 , we show the corresponding quantities for a constant step size $a = 0.1$.

We notice that, in most cases, we approach the optimal utility price from the non-feasible side ($f(t) > f^*$) because of the relaxation of the constrains. This means that the

[1]For simplicity, we can use one cvx call to solve all source problems at once since this is equivalent to solving them separately.

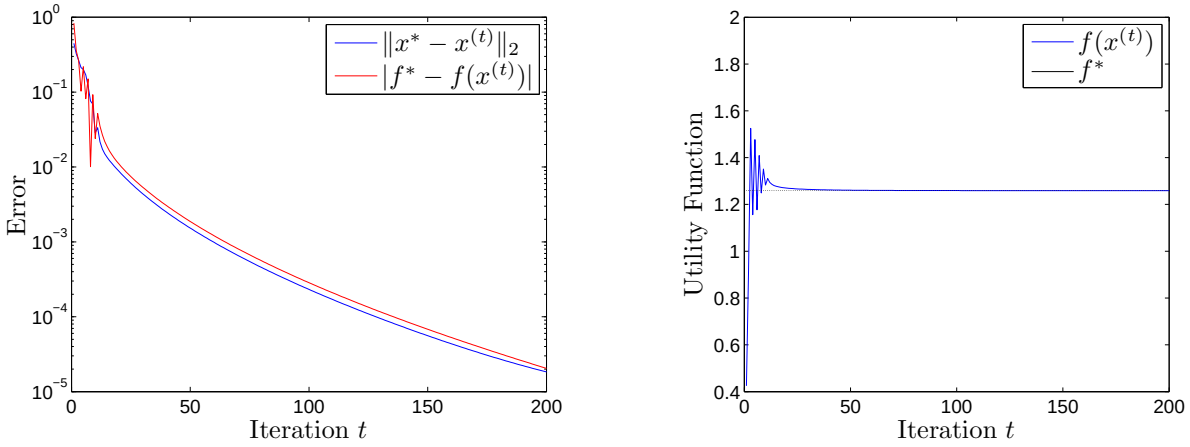

Figure 3.6: Dual Decomposition algorithm for the basic NUM problem using step $\alpha = 1/\sqrt{t}$. Left: In logarithmic scale we plot the error versus the iteration t. The blue line is the rate error $\|x^{(t)} - x^*\|_2$ and the red line is the absolute utility error $|f(x^{(t)}) - f^*|$. Right: the convergence of the utility function $f(x^{(t)})$ to the optimal price versus the iteration t in linear scale.

network is trying to work over its limits by not having zero residual on the constraints. So, we try to change the pricing faster when the constrains are violated. We propose to modify (3.15) as follows:

$$\lambda_l(t+1) = [\lambda_l(t) - \alpha\,(c_l - Ax)]^+ - \min(c_l - Ax, 0). \tag{3.20}$$

We have observed that, using this update, we reach feasible values faster. The results are shown in Figure 3.8, where we show the same quantities. The algorithm shown in Figure 3.7 is shown here for comparison with dotted line and we show the improved method with solid line.

3.3.3 Conclusions

- The basic NUM is one of the best ways to illustrate dual decomposition because the problem has a coupling constraint that decouples naturally (see (3.16)). A very important observation is as follows. There is no need for explicit message passing since $\lambda^{(s)}$ can be measured by each source s as the delay and $\sum_{s:l\in L(s)} x_s$ can be measured by each link l as the total traffic load, as we can see in Figure 3.9.

- The Dual decomposition method needs many iterations to converge, that is why we

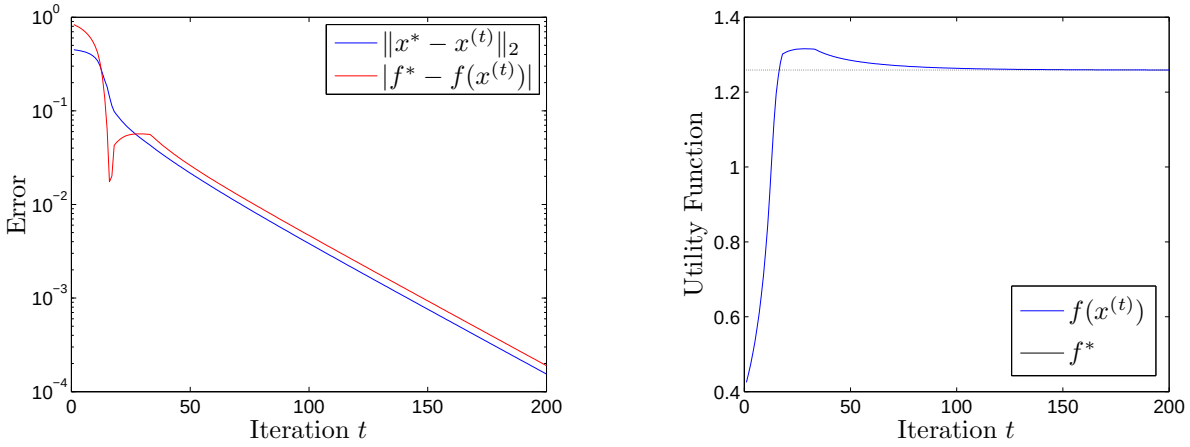

Figure 3.7: Dual Decomposition algorithm for the basic NUM problem using step $\alpha = 0.1$. Left: In logarithmic scale we plot the error versus the iteration t. The blue line is the rate error $\|x^{(t)} - x^*\|_2$ and the red line is the utility error $|f(x^{(t)}) - f^*|$. Right: the convergence of the utility function $f(x^{(t)})$ to the optimal price versus the iteration t in linear scale.

need to choose the step very carefully depending on the problem.

- The dual master problem is always convex regardless of the original problem. However, we still need convexity to have strong duality (under some constraint qualifications like Slater's condition).
- To solve the master problem, we can use different methods such as
  - bisection (if λ is scalar)
  - gradient or Newton method (if dual function is differentiable)
  - subgradient, cutting-plane, or ellipsoid method.
- The subproblems can be solved independently from master problem for a given λ.

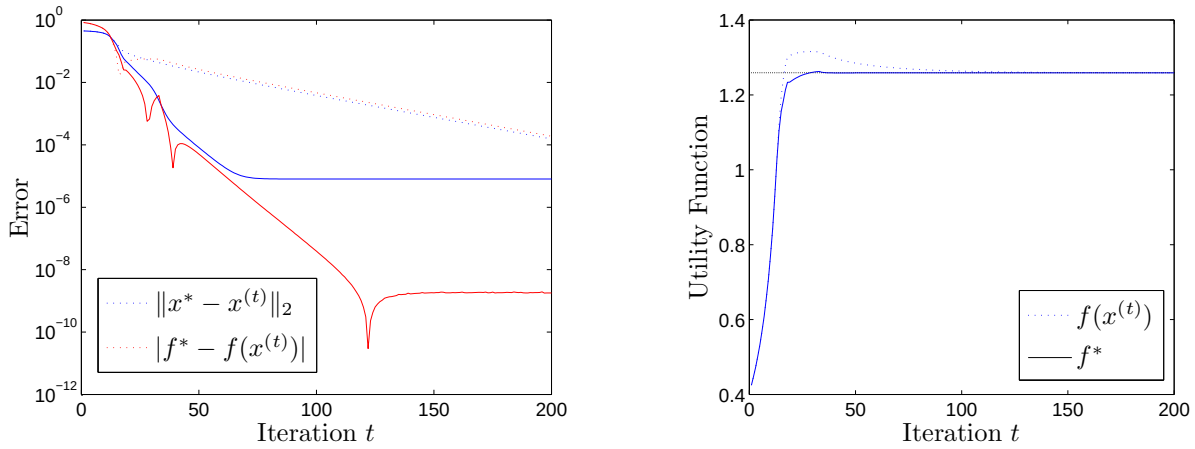

Figure 3.8: Dual Decomposition algorithm for the basic NUM problem using step $\alpha = 0.1$ (doted line) and "improved" step α for faster convergence from non feasible points (solid line). Left: In logarithmic scale we plot the error versus the iteration t. The blue line is the rate error $\|x^{(t)} - x^*\|_2$ and the red line is the absolute utility error $|f(x^{(t)}) - f^*|$. Right: the convergence of the utility function $f(x^{(t)})$ to the optimal price versus the iteration t in linear scale.

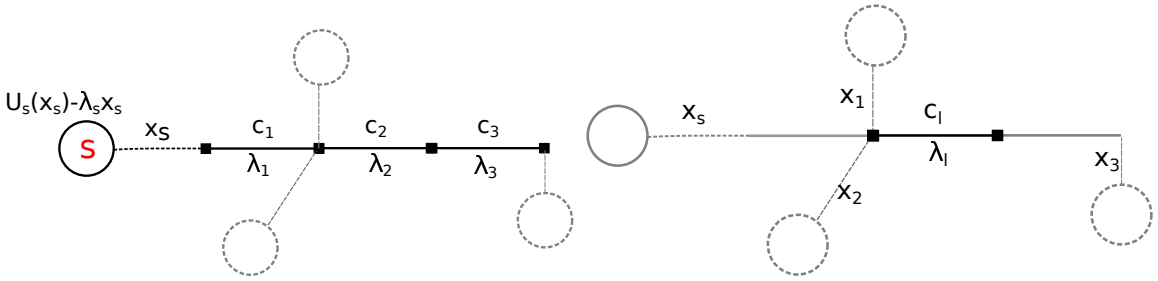

Figure 3.9: A network example from the perspective of a source (left) and a link (right).

Chapter 4

Alternative Decompositions

In Chapter 3, we illustrated the two basic methods of decomposition, Primal and Dual. We can combine these methods into multilevel decomposition to create many alternative decompositions for more complicated NUM problems. Multilevel decomposition is an important technique that leads to alternative distributed decompositions (Figure 4.1).

This chapter is based on [2], [17] and [18] and the efforts towards a systematic understanding of "layering as optimization decomposition," where the overall communication network is modelled as a generalized network utility maximization problem, each layer corresponds to a decomposed subproblem, and the interfaces among layers are quantified as functions of the optimization variables coordinating the subproblems. There can be many alternative decompositions, leading to a choice of different layering architectures.

There are three stages of conceptual understanding of an optimization/decomposition view of network architectures. First, layered and distributed network architectures can be rigorously understood as decompositions of an underlying optimization problem. Second, there are, in fact, many alternatives of decompositions and, therefore, alternatives of network architectures. Furthermore, we can systematically explore and compare such alternatives. Third, there may be a methodology to exhaustively enumerate all alternatives, to quantify various comparison metrics, and even to determine a priori which alternative is the best according to any given combination of comparison metrics. We will explore the second stage of the above list in this chapter, focusing on the algorithms rather than the network architecture.

We will start with a simple and popular extension of the basic NUM, which is the rate allocation problem with power constraints.

4.1 Power-Constrained Rate Allocation

In some applications, such as wireless broadcasting or Digital Subscriber Line (DSL) broadband access, distributed rate allocation can be carried out over transmission "pipes" of different sizes, with the help of adaptive resource allocation in the physical layer. This is an

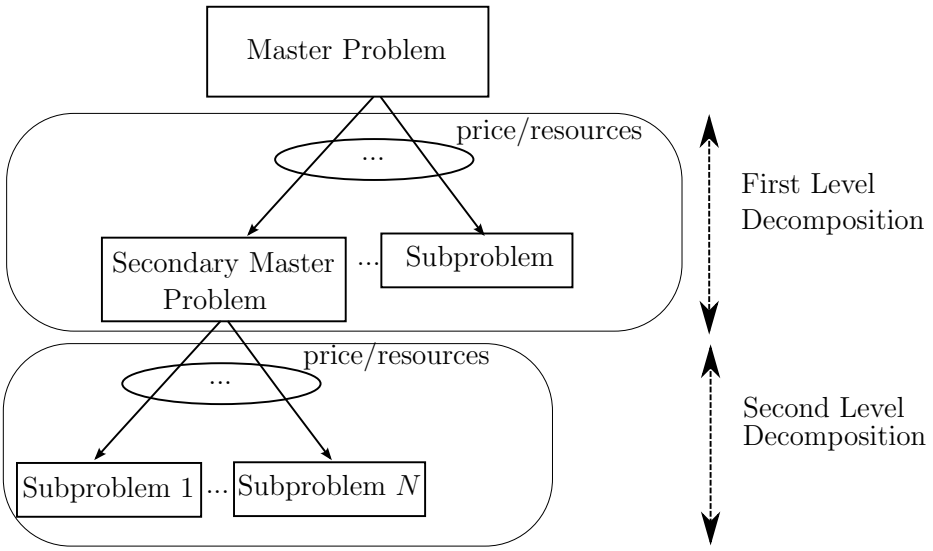

Figure 4.1: Multilevel decomposition

example of balancing "supply" of resources and "demand" of link capacities "built" from the limited resources. The algorithms in subsections 4.1.3 and 4.1.2 were first proposed in [20] and [21], respectively.

4.1.1 Problem Formulation

Consider the basic NUM in (3.12) but with variable link capacities $\{c_l(p_l)\}$, each of which depends on the allocated resource p_l, such as transmit power, with a constraint on the maximum total resource P_T associated with downlink transmissions. For many models, such as TDMA or FDMA, c_l is a strictly concave function.

Consider the following NUM formulation:

$$\begin{aligned} \underset{x,p\succeq 0}{\text{maximize}} \quad & U(x) = \sum_s U_s(x_s) \\ \text{subject to} \quad & \sum_{s:l\in L(s)} x_s \le c_l(p_l),\ \forall l \\ & \sum_l p_l \le P_T. \end{aligned} \tag{4.1}$$

Although it does not have many differences from the basic NUM, one can try different decompositions. We will consider two decompositions: a primal decomposition with respect to power allocation, and a dual decomposition with respect to the flow constraints.

Note that our problem formulation is rather general and different types of networks can be modelled in this way. For example, in wireless networks with Gaussian channels, the classical Shannon capacity formula gives that

$$c_l = W_l \log\left(1 + \frac{P_l}{\sigma_l W_l}\right) \tag{4.2}$$

where the adjustable parameters are W_l, the assigned bandwidth, and P_l, the power used in the link. Another case corresponds to distributed time sharing in a system operating under TDMA.

4.1.2 Primal-Dual Decomposition

Consider first a primal decomposition of (4.1) by fixing the power allocation. Clearly, the link capacities become fixed numbers and problem (4.1) becomes a basic NUM like (3.12), which can be solved via a dual decomposition as explained in Section 3.3. The master primal problem is

$$\begin{array}{ll} \underset{p \succeq 0}{\text{maximize}} & U^*(p) \\ \text{subject to} & \sum_l p_l \leq P_T \end{array} \tag{4.3}$$

where

$$U^*(p) = \max_{x \succeq 0} \{U(x) | Ax \preceq c(p)\}. \tag{4.4}$$

Note that $U^*(p)$ is the optimal objective value of (4.1) for a given p. Since a subgradient of $U^*(p)$ with respect to c_l is given by the Lagrange multiplier λ_l associated with the constraint $\sum_{s:l \in L(s)} x_s \leq c_l(p_l)$ in (4.1), using the following lemma, it follows that a subgradient of $U^*(p)$ with respect to p_l is given by $\lambda_l c_l'(p_l)$.

Lemma 4.1. *Under some mild assumptions, $c(p)$ is concave and a subgradient, $g(p)$, of $U^*(p)$ at p is given by*

$$g(p) = \left(\lambda_1 c_1'(p_1) \dots \lambda_L c_L'(p_L)\right)$$

where λ_l, for $l = 1, \dots, L$, are optimal Lagrange multipliers for the capacity constraints in (4.1) and $c_l'(p_l)$ is the derivative of function $c_l(\cdot)$ at p_l.

Proof. By strong duality,

$$\begin{aligned} U^*(p) &= \min_{\lambda \succeq 0} \max_{x} \left\{ \sum_{s=1}^{S} (U_s(x_s) - x_s q_s) + \sum_{l=1}^{L} \lambda_l c_l(p_l) \right\} \\ &= \min_{\lambda \succeq 0} \left\{ z(x(\lambda)) + \sum_{l=1}^{L} \lambda_l c_l(p_l) \right\} \end{aligned}$$

where (1) $q_s = \sum_{l=1}^{L} A_{l,s}\lambda_l$ and (2) z is the appropriate function. Since $U^*(p)$ is the pointwise infimum of concave functions, it is concave. Let λ^* be the optimal Lagrange multiplier vector for the resource allocation vector p. This implies that

$$U^*(p) = \max_{x} \left\{ \sum_{s=1}^{S} (U_s(x_s) - x_s q_s^*) \right\} + \sum_{l=1}^{L} \lambda_l^* c_l(p_l),$$

where $q_s^* = \sum_{l=1}^{L} A_{l,s}\lambda_l^*$.

For any other resource allocation $\tilde{p}$, it holds that

$$\begin{aligned} U^*(\tilde{p}) &\leq \max_{x} \left\{ \sum_{s=1}^{S} (U_s(x_s) - x_s q_s^*) + \sum_{l=1}^{L} \lambda_l^* c_l(\tilde{p}_l) \right\} \\ &= U^*(p) - \underbrace{\sum_{l=1}^{L} \lambda_l^* c_l(p_l)}_{f(p)} + \underbrace{\sum_{l=1}^{L} \lambda_l^* c_l(\tilde{p}_l)}_{f(\tilde{p})} \\ &\leq U^*(p) + \underbrace{\sum_{l=1}^{L} \lambda_l^* c_l'(p_l)(\tilde{p}_l - p_l)}_{\nabla f(\tilde{p})^T(\tilde{p}-p)} . \end{aligned}$$

The last inequality holds because functions $c_l(\cdot)$, for $l = 1, \ldots, L$ are concave. By the definition of the subgradient, this concludes the proof. □

Therefore, the master primal problem (4.3) can be solved with a subgradient method by updating the powers as

$$p(t+1) = \left[p(t) + \alpha \begin{bmatrix} \lambda_1^*(p(t))c_1'(p_1(t)) \\ \vdots \\ \lambda_L^*(p(t))c_L'(p_L(t)) \end{bmatrix} \right]_{\mathcal{P}}, \tag{4.5}$$

where $[.]_{\mathcal{P}}$ denotes the projection onto the feasible convex set $\mathcal{P} \triangleq \{p : p \succeq 0, \sum_l p_l \leq P_T\}$. Due to the projection, this subgradient update cannot be performed independently by each link and requires a centralized approach. The projection of a point (the expression inside the outer bracket in (4.5)) onto the simplex, can be easily obtained in the following waterfilling form:

$$p_l = (p_l^0 - \gamma)^+, \ \ \forall l, \tag{4.6}$$

where the waterlevel γ is chosen as the minimum nonnegative value such that $\sum_l p_l \leq P_T$. Observe that only the computation of γ requires a central node since the update of each power p_l can be done at each link.

4.1.3 Dual-Dual Decomposition

Consider now a dual decomposition of (4.1) by relaxing the flow constraints $\sum_{s:l\in L(s)} x_s \leq c_l(p_l)$. Using the Lagrange multipliers for the capacity constraints in (4.1) λ_l, for $l = 1, \ldots, L$, we form the partial Lagrangian

$$L(x, p, \lambda) = \left\{ \sum_s U_s(x_s) - \lambda^T (Ax - c(p)) \Big| \sum_l p_l = P_T \right\} \tag{4.7}$$

and the associated dual function

$$g(\lambda) = \sup_{x,p} L(x, p, \lambda) = \underbrace{\sup_x \left\{ \sum_s U_s(x_s) - \lambda^T A x \right\}}_{Network} + \underbrace{\sup_{\sum_l p_l = P_T} \lambda^T c(p))}_{Resource \ \ allocation} \tag{4.8}$$

This problem decomposes into one maximization for each source like the basic NUM in (3.12) (this is the network problem) plus the following maximization for the update of the power allocation

$$\begin{aligned} \underset{p\succeq 0}{\text{maximize}} \quad & \sum_l \lambda_l c_l(p_l) \\ \text{subject to} \quad & \sum_l p_l \leq P_T, \end{aligned} \tag{4.9}$$

which can be further decomposed, via a second-level dual decomposition, yielding the subproblems

$$\underset{p_l \geq 0}{\text{maximize}} \quad \lambda_l c_l(p_l) - \gamma p_l \tag{4.10}$$

whose solution is

$$\begin{aligned} \frac{\partial}{\partial p_l}\left\{\lambda_l c_l(p_l) - \gamma p_l\right\} = 0 \Rightarrow \\ c_l'(p_l) = \frac{\gamma}{\lambda_l} \Rightarrow \\ p_l = (c_l')^{-1}\left(\frac{\gamma}{\lambda_l}\right). \end{aligned} \tag{4.11}$$

We update the variable γ for the secondary master dual problem as

$$\gamma(t+1) = \left[\gamma(t) - \alpha\left(P_T - \sum_l p_l^*(\gamma(t))\right)\right]^+. \tag{4.12}$$

The master dual problem is solved as shown in Chapter 3 for the basic NUM.

4.1.4 Numerical Examples

We now demonstrate Primal-Dual and Dual-Dual Decomposition algorithms on a fixed network. The network has 8 nodes, and 12 direct links between nodes. The link capacities are given by

$$c_l(p_l) = \log(1 + p_l),$$

and the resource limit for the power is $P_T = 10$. The utility functions are $U_s(x_s) = cost_s \log(x_s)$, with random $cost_s \in \mathbb{R}^+$, which is very similar to proportional fairness. The example problem was solved with both algorithms explained in Subsections 4.1.2 and 4.1.3. The step lengths were tuned to approximately optimize the convergence rate for the two algorithms. In Figure 4.2, on the left, we plot in logarithmic scale the error $|U(x^{(t)}) - U^*|$ for both methods versus the iteration t. On the right, we can see how the utility function $U(x^{(t)})$ converges to the optimal price U^*.

4.1.5 Summary

We examined two ways for the distributed solution of power-constrained rate allocation problem in (4.1).

First, we examined the Primal Dual Decomposition method. We formed the master

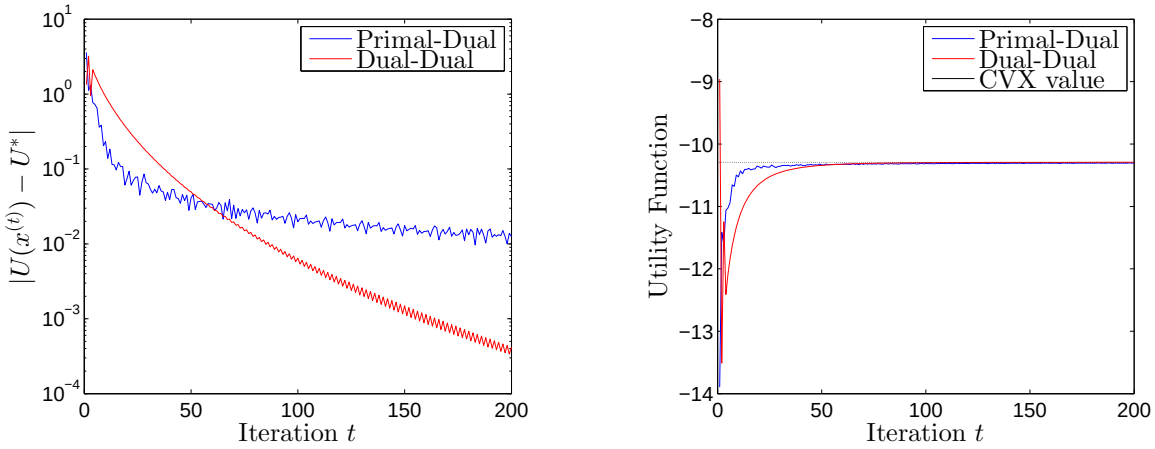

Figure 4.2: Primal-Dual Decomposition (blue line) and Dual-Dual Decomposition (red line) algorithms for NUM problem (4.1). Left: The error $|U(x^{(t)} - U^*|$ in logarithmic scale of the methods versus the iteration t. Right: The convergence of the utility function to the optimal price versus the iteration t in linear scale.

problem (4.3). In the t-th iteration we update

$$p_l^{(t+1)} := \left[p^{(t)} + \alpha^{(t)} \lambda_l^*(p^{(t)}) c_l'(p_l^{(t)})\right]_{\mathcal{P}}, \ l = 1, \ldots, L. \tag{4.13}$$

$$x_s^{(t+1)} := \arg\max_{x_s} \{U_s(x_s) - \lambda_{(s)}^{(t)} x_s\}, \ s = 1, \ldots, S. \tag{4.14}$$

$$\lambda_l^{(t+1)} := \left[\lambda_l^{(t)} - \alpha \left(c_l(p_l^{(t+1)}) - a_l^T x^{(t+1)}\right)\right]^+, \ l = 1, \ldots, L. \tag{4.15}$$

Second, we examined the Dual-Dual Decomposition method. In the t-th iteration we update

$$x_s^{(t+1)} := \arg\max_{x_s} \{U_s(x_s) - \lambda_{(s)}^{(t)} x_s\}, \ s = 1, \ldots, S. \tag{4.16}$$

$$\lambda_l^{(t+1)} := \left[\lambda_l^{(t)} - \alpha_\lambda^{(t)} \left(c_l(p_l^{(t+1)}) - a_l^T x^{(t+1)}\right)\right]^+, \ l = 1, \ldots, L. \tag{4.17}$$

$$p_l^{(t+1)} := (c_l')^{-1} \left(\frac{\gamma^{(t)}}{\lambda_l^{(t)}}\right), \ l = 1, \ldots, L. \tag{4.18}$$

$$\gamma^{(t+1)} := \left[\gamma^{(t)} - \alpha_\gamma^{(t)} \left(P_T - \sum_l p_l^{(t+1)}\right)\right]^+. \tag{4.19}$$

$$\tag{4.20}$$

4.2 QoS Rate Allocation

4.2.1 Problem Formulation

Consider the basic NUM but with different classes of users that will be treated differently. We constrain the rates to be within a range for each class. In our example, we assume only two classes of users. Denoting by $y_l^{(1)}$ and $y_l^{(2)}$ the aggregate rates of classes 1 and 2, respectively, along the l_{th} link the problem formulation is

$$\begin{aligned} \underset{x,y^{(1)},y^{(2)}\succeq 0}{\text{maximize}} \quad & U(x) = \sum_s U_s(x_s) \\ \text{subject to} \quad & \sum_{s\in S_i: l\in L(s)} x_s \le y_l^{(i)}, \quad \forall l, i = 1,2 \\ & y^{(1)} + y^{(2)} \preceq c \\ & c_{\min}^{(i)} \preceq y^{(i)} \preceq c_{\max}^{(i)}, \ \forall l, i = 1,2. \end{aligned} \tag{4.21}$$

The numbers $c_{\min}^{(i)}$ and $c_{\max}^{(i)}$ are the limits of the aggregate rate for each class applied in every link of the network. The constraints $c_{\min}^{(i)} \le y^{(i)} \le c_{\max}^{(i)}$ are the Quality of Service (QoS) requirements of the network. Observe that in the absence of these constraints, problem (4.21) becomes the basic NUM. Also note that, without loss of generality, the equality flow constraints can be rewritten as inequality flow constraints.

We will consider two decompositions: a *primal decomposition* with respect to the aggregate rate of each class, and a *dual decomposition* with respect to the total aggregate rate constraints from both classes.

4.2.2 Primal-Dual Decomposition

Consider first a primal decomposition of (4.21) by fixing the aggregate rates $y^{(1)}$ and $y^{(2)}$. Then, problem (4.21) is decomposed into two *independent* basic NUMs, for $i = 1, 2$,

$$\begin{aligned} \underset{x\succeq 0}{\text{maximize}} \quad & \sum_{s\in S_i} U_s(x_s) \\ \text{subject to} \quad & \sum_{s\in S_i: l\in L(s)} x_s \le y_l^{(i)}, \quad \forall l, \end{aligned} \tag{4.22}$$

where the fixed aggregate rates $y_l^{(i)}$ play the role of the fixed link capacities in the basic NUM. Each of these two subproblems can be solved like a basic NUM. The master problem

is

$$\begin{aligned} \underset{y^{(1)},y^{(2)}\succeq 0}{\text{maximize}} \quad & U_1^*(y^{(1)}) + U_2^*(y^{(2)}) \\ \text{subject to} \quad & y^{(1)} + y^{(2)} \preceq c \\ & c_{\min}^{(i)} \preceq y^{(i)} \preceq c_{\max}^{(i)},\ i = 1, 2, \end{aligned} \tag{4.23}$$

where $U_i^*(y^{(i)})$ is the optimal objective value of the problem (4.22) for the i_{th} class for a given $y^{(i)}$. Each link updates locally the aggregates rates $y^{(i)}$ and the set of different prices $\lambda^{(i)}$ for each QoS class using the subgradient algorithm. The master primal problem can now be solved by updating the aggregate rates as

$$\begin{bmatrix} y^{(1)}(t+1) \\ y^{(2)}(t+1) \end{bmatrix} = \left[\begin{bmatrix} y^{(1)}(t) \\ y^{(2)}(t) \end{bmatrix} + \alpha \begin{bmatrix} \lambda^{*(1)}(y^{(1)}(t)) \\ \lambda^{*(2)}(y^{(2)}(t)) \end{bmatrix} \right]_y \tag{4.24}$$

where $[.]_y$ denotes the projection onto feasible convex set.

4.2.3 Partial Dual Decomposition

Consider now a Dual Decomposition of problem (4.21) by relaxing only the flow constraints $\sum_{s\in S_i : l\in L(s)} x_s \leq y_l^{(i)},\ l, i = 1, 2$. This problem decomposes into

- one maximization per source as the maximization of basic NUM,

- one additional maximization to update the aggregate rates:

$$\begin{aligned} \underset{y^{(1)},y^{(2)}\succeq 0}{\text{maximize}} \quad & \lambda^{(1)T} y^{(1)} + \lambda^{(2)T} y^{(2)} \\ \text{subject to} \quad & y^{(1)} + y^{(2)} \preceq c \\ & c_{\min}^{(i)} \preceq y^{(i)} \preceq c_{\max}^{(i)},\ i = 1, 2, \end{aligned} \tag{4.25}$$

which can be solved independently by each link with knowledge of the corresponding Lagrange multipliers $\lambda_l^{(1)}$ and $\lambda_l^{(2)}$, which in turn are also updated independently by each link.

The master dual problem corresponding to this dual decomposition is updated with the following subgradient method:

$$\lambda_l^{(i)}(t+1) = \left[\lambda_l^{(i)}(t) - \alpha\left(y_l^{(i)}(t) - \sum_{s\in S_i: l\in L(s)} x_s^*\left(\lambda^{(i)s}(t)\right)\right)\right]^+, \quad i = 1, 2. \tag{4.26}$$

4.2.4 Numerical Examples

We implemented both algorithms in Matlab for many sources and link sizes and also many sparse or dense matrices A. For our experiments, we used the concave source utility functions $U_s(x_s) = cost_s \log x_s$, with random $cost_s \in \mathbb{R}^+$, for $s = 1, \ldots, S$. We randomize our QoS limits so there is a feasible point and we partition sources into classes. We randomize our incidence matrix A in a way that we can control the sparsity. We solve the dual problems with `cvx` for each source. We could solve them iteratively for each source like the basic NUM in Section 3.3.

In the following Figures, we show Primal-Dual and Partial Dual Decomposition algorithms. Left, we plot the error $|U(x^{(t)}) - U^*|$ in logarithmic scale, and in the right we plot the convergence of the utility function $U(x^{(t)})$ to the optimal value U^* for both algorithms. Note that the optimal value is computed by `cvx`. We present the following cases.

- In Figure 4.3, we show both methods for $Links = 15$, $Sources = 5$ and 50% average link use per source. We use constant step size $\alpha = 0.1$ for both methods.
- In Figure 4.4, we show both methods for $Links = 15$, $Sources = 5$ and 10% average link use per source $\alpha = 0.4$ for primal-dual and $\alpha = 0.01$ for partial-dual[1].
- In Figure 4.5, we show both methods for $Links = 150$, $Sources = 50$ and 5% average link use per source $\alpha = 0.1$ for primal-dual and $\alpha = 0.01$ for partial-dual. In Figure 4.6, we use double step size for partial-dual algorithm when subgradient is negative so it converges faster from non-feasible area.

4.2.5 Conclusions

Summarizing, we examined two different methods for distributed solutions for rate allocation among QoS classes in (4.21).

[1]This algorithm is very sensitive to big changes and it doesn't always converge with big size steps.

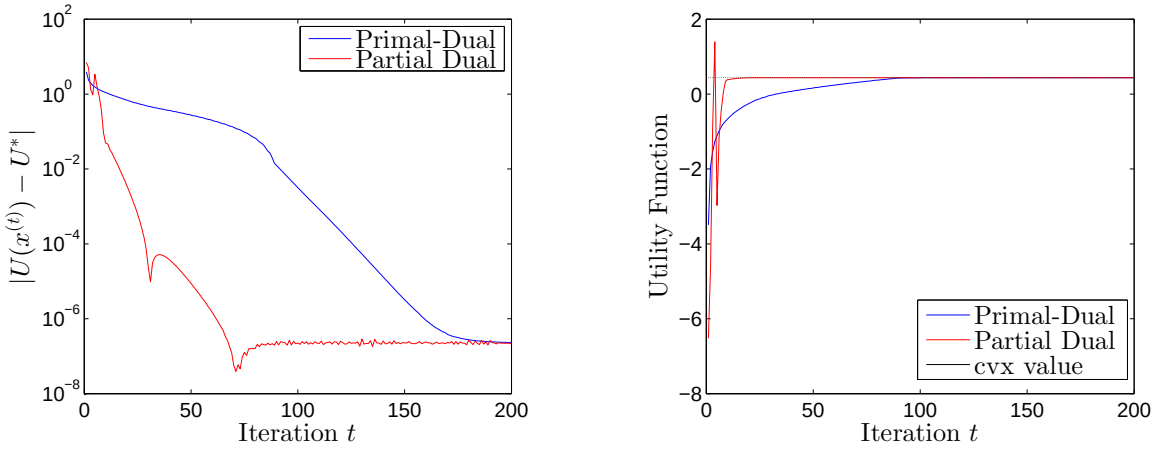

Figure 4.3: Primal-Dual (blue line) and Partial Dual (red line) Decomposition methods for constant step size and 50% link use per source. Left: The error $|U(x^{(t)}) - U^*|$ in logarithmic scale versus t. Right: the convergence of the utility function to the optimal price in linear scale.

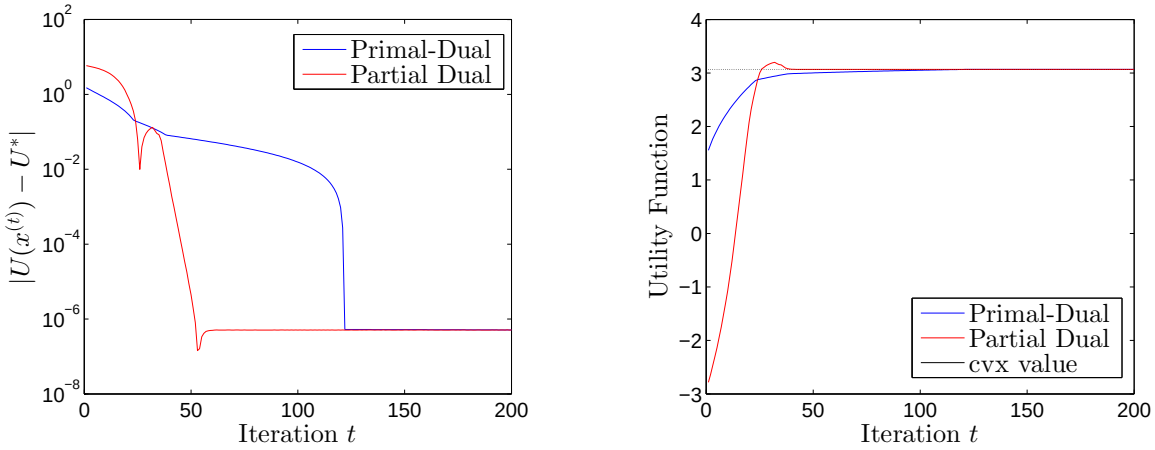

Figure 4.4: Primal-Dual (blue line) and Partial Dual (red line) Decomposition methods for constant step size and 10% link use per source. Left: The error $|U(x^{(t)}) - U^*|$ in logarithmic scale versus t. Right: the convergence of the utility function to the optimal price in linear scale.

First, we examined the Primal-Dual Decomposition method. We formed the master

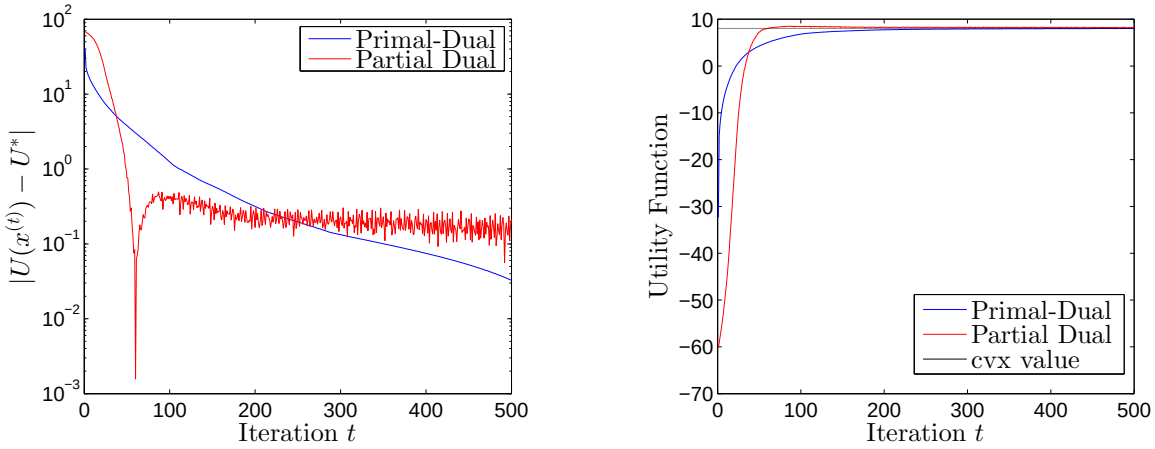

Figure 4.5: Primal-Dual (red line) and Partial Dual (green line) Decomposition methods for constant step size, 50 sources, 150 links, and 5% link use per source. Left: The squared error in logarithmic scale versus the iterations. Right: the convergence of the utility function to the optimal price in linear scale.

problem (4.23). In the t-th iteration we update

$$x_s^{(t+1)} := \arg\max_{x_s}\{U_s(x_s) - \lambda_{(s)}^{(i)(t)} x_s\},\ s = 1, \ldots, S,\ i = 1, 2. \tag{4.27}$$

$$\lambda_l^{(i)(t+1)} := \left[\lambda_l^{(i)(t)} - \alpha_\lambda^{(t)}\left(y^{(i)(t+1)}\right) - a_l^T x^{(t+1)}\right)\Big]^+,\ l = 1, \ldots, L,\ i = 1, 2. \tag{4.28}$$

$$y^{(i)(t+1)} := \left[y^{(i)(t)} - \alpha_y^{(t)} \lambda_l^{(i)(t+1)}\right]_y,\ i = 1, 2. \tag{4.29}$$

Second, we examined the Partial Dual Decomposition method. In the t-th iteration we update

$$x_s^{(t+1)} := \arg\max_{x_s}\{U_s(x_s) - \lambda_{(s)}^{(t)} x_s\},\ s = 1, \ldots, S. \tag{4.30}$$

$$y_l^{(1,2)(t+1)} := \arg \max_{y_l^{(1)}, y_l^{(2)} \geq 0} \left\{ \begin{array}{c} \lambda_l^{(1)(t)} y_l^{(1)} + \lambda_l^{(2)(t)} y_l^{(2)} \\ \text{s.t. } y^{(1)} + y^{(2)} \preceq c, \\ c_{\min}^{(i)} \preceq y^{(i)} \preceq c_{\max}^{(i)} \end{array} \right\},\ l = 1, \ldots, L. \tag{4.31}$$

$$\lambda_l^{(i)(t+1)} := \left[\lambda_l^{(i)(t)} - \alpha_\lambda^{(t)} \left(y_l^{(i)(t+1)} - \sum_{s \in S_i : l \in L(s)} x_s^{(t+1)} \right) \right]^+,\ l = 1, \ldots, L,\ i = 1, 2. \tag{4.32}$$

Observations:

- Partial-Dual algorithm has one level of decomposition, Primal-Dual algorithm has

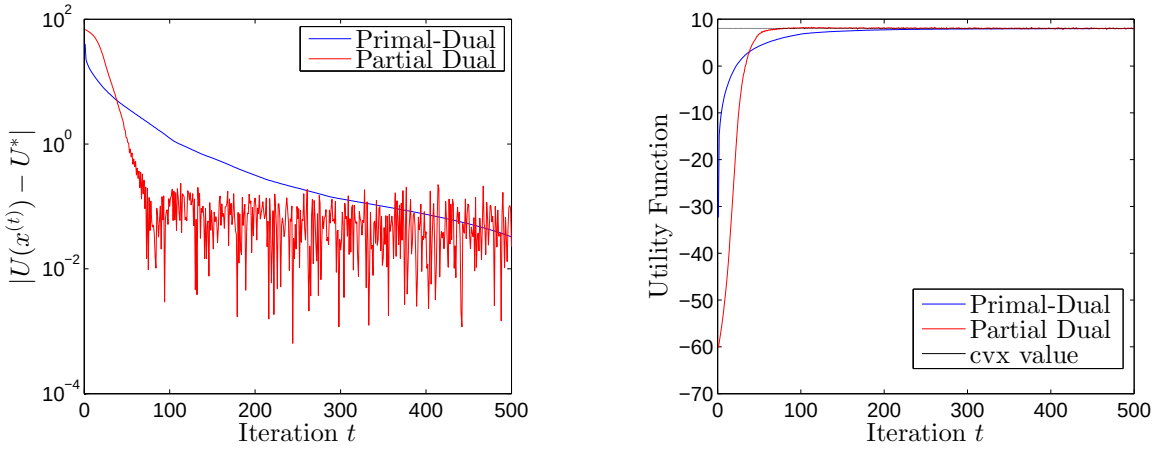

Figure 4.6: Primal-Dual (red line) and Partial Dual (green line) Decomposition methods for constant step size, 50 sources, 150 links, and 5% link use per source. We use an "improved" Partial Dual algorithm with double step size when we are in non feasible point. Left: The squared error in logarithmic scale of the method versus the iterations. Right: the convergence of the utility function to the optimal price in linear scale.

two levels of decomposition. Both algorithms do not need signalling between nodes to work.

- To solve the master problem, both algorithms use subgradient method.
- Primal-Dual is not that fast as the figures show because there are the NUM subproblems that must be solved iteratively if we want to solve them in a distributed manner.
- The sparsity of matrix A affects algorithm convergence. As long as the average amount of links each source uses stays the same, the independent NUMs will converge in the same way regardless how many other sources are in the network, and it will not affect the maximum number of iterations needed for the algorithm to converge. This means that if we have a big number of sources that only use in average k numbers of links each, then the algorithm will converge.
- In the primal-dual decomposition approach, each link updates the aggregate rates on a slower timescale and the prices on a faster timescale whereas, in partial dual decomposition approach, each link updates the price on a slower timescale and the aggregate rates on a faster timescale. Therefore, the speed of convergence of the partial dual approach should be faster, in general. In both cases, the users are

ignorant of the existence of classes and only the links have to take this into account by having one price for each class. In other words, this is a way to give each class of users a different price than the one based on the standard dual-based algorithm so that they can be differentiated into different QoS classes. The next application hinges on this observation.

4.3 Hybrid Rate-Based and Price-Based Rate Allocation

One extreme way to control the rate allocation process is to directly give each source the rate they can use, at the expense of a centralized computation. At the other extremum, we can optimize the system in a fully distributed way via pricing feedback, as in the basic NUM, at the expense of trusting the sources even though they can be noncooperative and try to obtain more bandwidth by using a more aggressive utility function. Neither of these two extreme approaches is completely satisfactory in all applications, and hybrid solutions between rate-based and window-based rate allocation are desirable for both robustness of fair allocation against aggressive users and speed of converging to the correct rate allocation equilibrium.

New congestion control protocols using direct rate allocation have recently been proposed, such as eXplicit Control Protocol (XCP) in [22] and Rate Control Protocol (RCP) in [23], which are based on a heuristic computation of the processor-sharing type of rate allocation by each router that a flow traverses. We now describe a systematic method using primal decomposition to perform distributed and direct rate allocation to each user. It turns out that direct rate control is a special case of alternative NUM decompositions.

4.3.1 Problem Formulation

The key idea is to use the approach of Section 4.2 but with one class for each user. The NUM formulation becomes

$$\begin{aligned} \underset{x,y^{(s)}\succeq 0}{\text{maximize}} \quad & \sum_s U_s(x_s) \\ \text{subject to} \quad & x_s \le y_l^{(s)}, \quad \forall s, l \in L(s) \\ & \sum_s y_l^{(s)} \le c_l \\ & c_{\min}^{(s)} \le y_l^{(s)} \le c_{\max}^{(s)}, \ \forall s, l \in L(s). \end{aligned} \tag{4.33}$$

Note that if a source s does not use a path l, then $y_l^{(s)}$ is taken as zero in the constraint $\sum_s y_l^{(s)} \le c_l$.

4.3.2 Primal Decomposition

If we take a primal decomposition approach, then the master primal problem will be in charge of the update of $y_l^{(s)}$ and each user will simply choose x_s equal to the minimum of the $y_l^{(s)}$ along its path in order to satisfy $x_s \le y_l^{(s)}, \ \forall s, \ l \in L(s)$. This approach constitutes one of the extreme methods in which each user is directly given the amount of bandwidth it can use.

4.3.3 Partial Dual Decomposition

We may also take a dual decomposition approach by relaxing the flow constraints

$$\begin{aligned} \underset{x,y^{(s)}\succeq 0}{\text{maximize}} \quad & \sum_s \left[U_s(x_s) - \left(\sum_{l\in L(s)} \lambda_l^{(s)} \right) x_s \right] + \sum_s \sum_{l\in L(s)} \lambda_l^{(s)} y_l^{(s)} \\ \text{subject to} \quad & \sum_s y_l^{(s)} \le c_l, \forall l \\ & c_{\min}^{(s)} \le y_l^{(s)} \le c_{\max}^{(s)}, \ \forall s, l \in L(s). \end{aligned} \tag{4.34}$$

This problem decomposes into one maximization for each source, with $\lambda^s = \sum_{l\in L(s)} \lambda_l^{(s)}$ being the aggregate path price specific for user s, plus the following additional rate-bounding

maximization to obtain the $y_l^{(s)}$, for each link l,

$$\begin{aligned} \underset{y^{(s)} \succeq 0}{\text{maximize}} \quad & \sum_{s:l \in L(s)} \lambda_l^{(s)} y_l^{(s)} \\ \text{subject to} \quad & \sum_s y_l^{(s)} \leq c_l, \ \forall l \\ & c_{\min}^{(s)} \leq y_l^{(s)} \leq c_{\max}^{(s)}, \ \forall s, l \in L(s). \end{aligned} \tag{4.35}$$

This problem can be solved independently by each link as a way to distribute its capacity c_l among the sources using the link according to the weights given by the prices $\lambda_l^{(s)}$, which are different for each source.

$$\lambda_l^{(s)}(t+1) = [\lambda_l^{(s)}(t) - \alpha(y_l^{(s)}(t) - x_s^*(\lambda_l^s(t))]^+ \quad \forall l, s : l \in L(s), \tag{4.36}$$

where $x_s^*(\lambda_l^s(t))$ is the optimal rate x_s for source s for the maximization derived form (4.34) for a given λ_l^s at a time t.

4.3.4 Numerical Example

We tested the above algorithms in a network with $sources = 5$ and $links = 10$. The utility function was $U(x_s) = cost_s \log(x_s)$ for each source. We used an average of 30% link use per source.

In Figure 4.7, we show with red line the Primal-Dual Decomposition algorithm and with the blue line the Primal Decomposition algorithm. In the left plot we show, in logarithmic scale, the error of the algorithm by showing the quantity $|U(x^{(t)}) - U^*|$ versus the iteration t. In the right plot of Figure 4.7, we show, in linear scale, the utility function $U(x^{(t)})$ versus iteration t, plus the optimal value we have from the software `cvx`. In Figure 4.8, we show the corresponding quantities for different step sizes. In General, the Partial-Dual decomposition is faster than primal decomposition. But we observe Primal decomposition is more stable. In Figure 4.8, we can see a similar convergence between the two algorithms, achieved with the proper parameters like step sizes.

4.3.5 Summary

Summarizing, we examined two different methods for the solution of Hybrid Rate-Based and Price-Based Rate Allocation problem of (4.33)

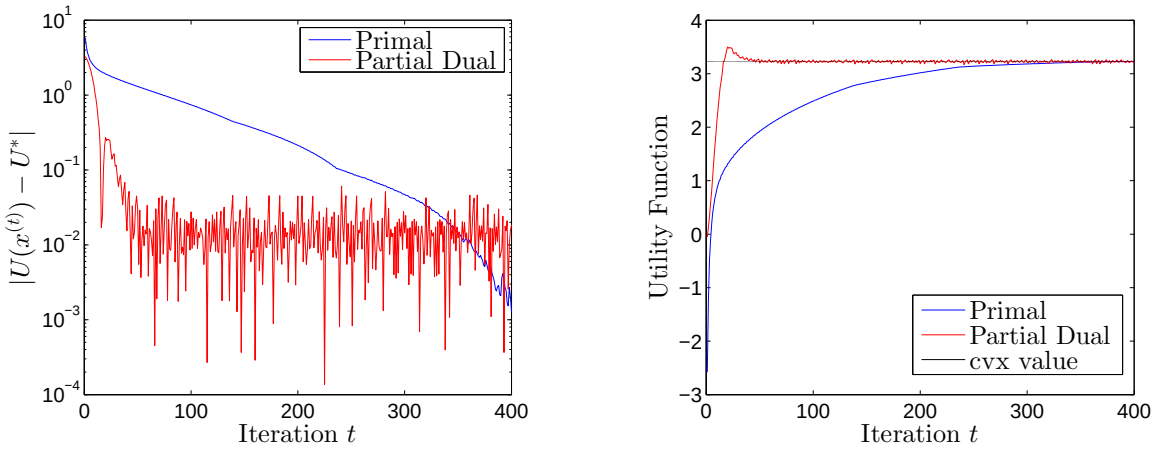

Figure 4.7: Primal and Primal-Dual Decomposition methods for Hybrid rate-based and price based rate allocation for step sizes $a_1 = a_2 = 0.1$. Left: The error $|U(x^{(t)}) - U^*|$ versus t. Right: the utility function $U(x^{(t)})$ versus t.

Primal Decomposition leads to a direct rate allocation and is based on one level decomposition. In the t-th iteration we update

$$x_s^{(t+1)} := \min_{s:l\in L(s)} \{y_l^{(s)}\},\ s = 1, \ldots, S, \tag{4.37}$$

$$\lambda_l^{(s)(t+1)} := \left[\lambda_l^{(s)(t)} - \alpha_\lambda^{(t)} \left(y_l^{(s)(t+1)} - \sum_{s\in S:l\in L(s)} x_s^{(t+1)}\right)\right]^+,\ l = 1, \ldots, L. \tag{4.38}$$

$$y^{(s)(t+1)} := \left[y^{(s)(t)} - \alpha_y^{(t)} \lambda_l^{(s)(t+1)}\right]_y,\ i = 1, 2. \tag{4.39}$$

This approach requires the signalling to inform each user what rate to transmit at.

Partial Dual Decomposition method only shows one level of decomposition and does not require any explicit signalling. It is a hybrid rate-bounding and pricing feed-back

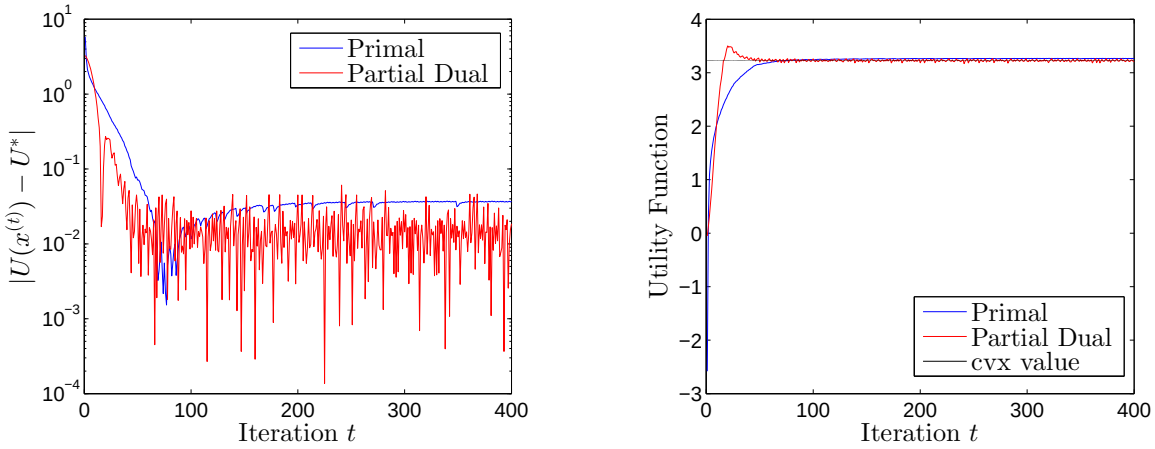

Figure 4.8: Primal and Primal-Dual Decomposition methods for Hybrid rate-based and price based rate allocation for step sizes $a_1 = 0.5$ and $a_2 = 0.01$ respectively. Left: The error $|U(x^{(t)}) - U^*|$ versus t. Right: the utility function $U(x^{(t)})$ versus t.

mechanism. In the t-th iteration we update

$$x_s^{(t+1)} := \arg\max_{x_s} \{U_s(x_s) - \lambda_{(s)}^{(t)} x_s\},\ s = 1, \ldots, S. \tag{4.40}$$

$$y_l^{(s)(t+1)} := \arg\max_{y_l^{(s)} \geq 0} \left\{ \begin{array}{l} \sum_{s:l \in L(s)} \lambda_l^{(s)(t)} y_l^{(s)} \\ \text{s.t. } \sum_s y_l^{(s)} \leq c_l, \\ c_{\min}^{(s)} \preceq y_l^{(s)} \preceq c_{\max}^{(s)} \end{array} \right\},\ l = 1, \ldots, L. \tag{4.41}$$

$$\lambda_l^{(s)(t+1)} := \left[\lambda_l^{(s)(t)} - \alpha_\lambda^{(t)} \left(y_l^{(s)(t+1)} - \sum_{s \in S_i : l \in L(s)} x_s^{(t+1)} \right) \right]^+,\ l = 1, \ldots, L. \tag{4.42}$$

4.4 Multipath-Routing Rate Allocation

We now consider a more general setup of the basic NUM illustrated in Chapter 3 where each source can choose among several possible paths and possibly use a weighted combination of them.

A similar formulation of multipath routing utility maximization is considered in [24], where a proximal optimization method followed by full dual decomposition is taken. This type of multi-path utility maximization problems appear naturally in several resource allocation problems in communication networks, such as the multi-path flow control problem, the optimal QoS routing problem, and the optimal network pricing problem.

4.4.1 Problem Formulation

The structure of a network with S sources, L links, and J paths can be summarized by the $L \times S$ path availability $0-1$ matrix H defined at

$$H_{l,j} := \begin{cases} 1, & \text{if the } j_{th} \text{ path uses the } l_{th} \text{ link,} \\ 0, & \text{otherwise,} \end{cases}$$

together with the $J \times S$ path choice nonnegative matrix W

$$W_{j,s} = \begin{cases} w_{js}, & \text{if the } s_{th} \text{ source uses the } j_{th} \text{ path,} \\ 0, & \text{otherwise,} \end{cases}$$

where w_{js} indicates the percentage of the rate of the s_{th} user allocated to the j_{th} path and has to satisfy $w_{js} > 0$ and $\sum_j w_{js} = 1$. These two matrices can be combined into the routing matrix

$$R = HW \tag{4.43}$$

that tells how much each source is using each link. Note that this notation of matrices H and W follows that in [25]. However, the problem being considered here is to design a rate allocation algorithm with fixed H and W, whereas the problem considered in [25] is to analyze the effect of joint routing and rate allocation with W being a variable.

The problem can be formulated with the routing matrix R like the basic NUM in (3.13)

$$\begin{aligned} \underset{x \succeq 0}{\text{maximize}} \quad & \sum_s U_s(x_s) \\ \text{subject to} \quad & Rx \preceq c \end{aligned} \tag{4.44}$$

and then the standard dual-based decomposition algorithm can be used. In this section, we will formulate the problem alternatively in terms of H and W as follows:

$$\begin{aligned} \underset{x \succeq 0}{\text{maximize}} \quad & \sum_s U_s(x_s) \\ \text{subject to} \quad & Wx \preceq y \ \text{ (path constraint)} \\ & Hy \preceq c \ \text{ (link constraint)} \end{aligned} \tag{4.45}$$

where y_l contains the aggregate rate along the l_{th} path.

4.4.2 Primal-Dual Decomposition

For a fixed y, problem (4.45) becomes the basic NUM problem, that can be solved with a dual decomposition approach and we can update the path rates y with a subgradient method. So, the master primal problem is

$$\begin{aligned} \underset{y \succeq 0}{\text{maximize}} \quad & U^*(y) \\ \text{subject to} \quad & Hy \preceq c, \end{aligned} \tag{4.46}$$

where $U^*(y)$ is the optimal objective value of (4.45) for a given y. The subgradient is given by the Lagrange multiplier λ associated to the constrains $Wx \leq y$ in (4.45). So we update the path rates as follows

$$y(t+1) = \left[y(t) + \alpha \lambda^*(y(t)) \right]_{\mathcal{Y}}, \tag{4.47}$$

where $[.]_{\mathcal{Y}}$ denotes the projection onto the feasible convex set $\mathcal{Y} \triangleq \{y : y \geq 0, Hy \leq c\}$. Note that this subgradient update cannot be performed independently by each path due to the projection onto $\mathcal{Y}$, which makes it impractical.

4.4.3 Partial Dual Decomposition

We will also present a method similar to [26]. We take a partial dual decomposition of (4.45) by relaxing only the constraint $Wx \preceq y$,

$$\begin{aligned} \underset{x,y\succeq 0}{\text{maximize}} \quad & \sum_s U_s(x_s) + \gamma^T(y - Wx) \\ \text{subject to} \quad & Hy \preceq c. \end{aligned} \tag{4.48}$$

This problem decomposes into one maximization for the sources, like the one in Section 3.3 for the basic NUM,

$$\underset{x\succeq 0}{\text{maximize}} \quad \sum_s \left[U_s(x_s) - \gamma^s x_s\right], \tag{4.49}$$

where $\gamma^s = \gamma^T W_{:,s} = \sum_{j\in J(s)} \gamma_j w_{js}$ is the aggregate price for the s-th source, plus one maximization for the path rates

$$\begin{aligned} \underset{y\succeq 0}{\text{maximize}} \quad & \gamma^T y \\ \text{subject to} \quad & Hy \preceq c. \end{aligned} \tag{4.50}$$

which has to be solved in a centralized way.

The master dual problem updates the prices as

$$\gamma(t+1) = \left[\gamma(t) - \alpha(y - Wx(\gamma(t)))\right]^+. \tag{4.51}$$

4.4.4 Full Dual Decomposition

Another way to solve problem (4.45) is with full dual decomposition by relaxing both constraints $Wx \preceq y$ and $Hy \preceq c$

$$\underset{x\succeq 0,y}{\text{maximize}} \quad \sum_s U_s(x_s) + \gamma^T(y - Wx) + \lambda^T(c - Hy) \tag{4.52}$$

which can be rewritten as

$$\underset{x\succeq 0,y}{\text{maximize}} \quad \sum_s [U_s(x_s) - x_s\gamma^{(s)}] + \sum_j y_j(\gamma_j - \lambda^{(j)}) + \lambda^T c \tag{4.53}$$

where $\lambda^{(j)} = \lambda^T H_{:,j} = \sum_{l \in J(j)} \lambda_l$ is the aggregate price of the j_{th} path and $\gamma^{(s)} = \gamma^T W_{:,s} = \sum_{j \in J(s)} \gamma_j w_{js}$ is the aggregate price for the s_{th} source. This problem decouples into maximization over x, like the basic NUM, and maximization over y, which is unbounded unless $\gamma_j = \lambda^{(j)}$. Therefore the optimal choice for the master dual problem is $\gamma_j = \lambda^{(j)}$ and then $\gamma^{(s)} = \sum_{j \in J(s)} \lambda^{(j)} w_{js} = \sum_{j \in J(s)} w_{js} \sum_{l \in J(j)} \lambda_l$. Hence, this approach reduces to the standard dual-based algorithm applied to problem (4.44)

4.4.5 Numerical Example

We now consider a NUM with different groupings of the path and link constraints as described in this Section. In particular, we generate a random network topology with $S = 4$ sources, $J = 12$ paths, and $L = 36$ links, such that each user uses three paths on average and each path uses five links on average.

In Figure 4.9, we show the Primal-Dual Decomposition algorithm. In the right plot we show, in logarithmic scale, the error of the algorithm by showing the quantity $|U(x^{(t)}) - U^*|$ versus the iteration t. In the left plot, we show in linear scale, the utility function $U(x^{(t)})$ versus iteration t, plus the optimal value we have from the software cvx. In Figure 4.10, we show the corresponding quantities for Partial-Dual Decomposition algorithm.

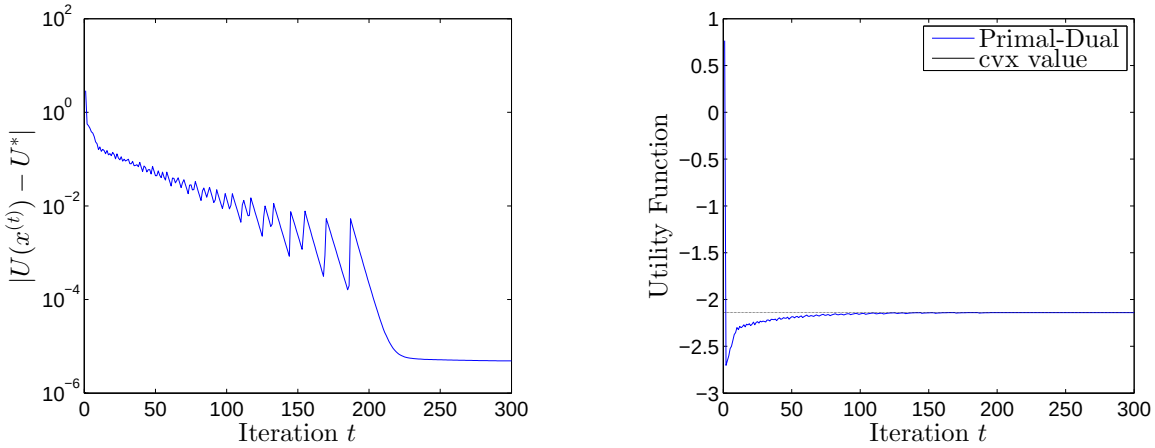

Figure 4.9: Primal-Dual Decomposition for multipath-routing rate allocation problem with step $\alpha_t = 0.01$. Left: The error $|U(x^{(t)}) - U^*|$ in logarithmic scale. Right: the utility function in linear scale.

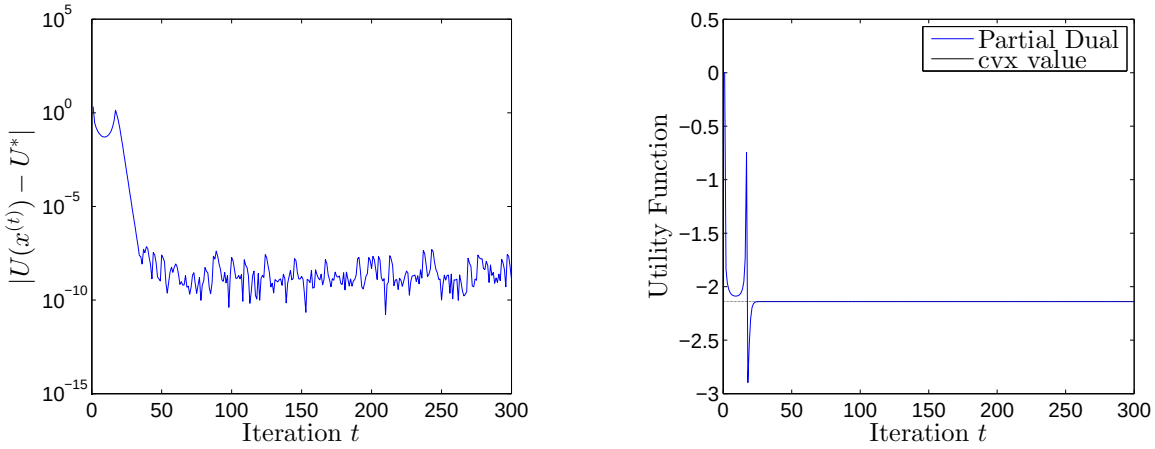

Figure 4.10: Partial-Dual Decomposition for multipath-routing rate allocation problem with Polyak's step (Section2.3.3). Left: The error $|U(x^{(t)}) - U^*|$ in logarithmic scale. Right: the utility function in linear scale.

4.4.6 Summary

Summarizing, we examined several different methods for the solution of Multipath-Routing Rate Allocation problem of (4.45)

Primal-Dual Decomposition. In the t-th iteration we update

$$x_s^{(t+1)} := \min_{x_s}\{U_s(x_s)|(Wx) \leq y^{(t)}\},\ s = 1, \ldots, S, \tag{4.54}$$

$$\lambda^{(s)(t+1)} := \left[\lambda^{(s)(t)} - \alpha_\lambda^{(t)}(y^{(s)(t+1)} - Wx^{(t)})\right]^+,\ l = 1, \ldots, L. \tag{4.55}$$

$$y^{(t+1)} = \left[y^{(t)} + \alpha\lambda^{(t)}\right]_{\mathcal{Y}}, \tag{4.56}$$

Unfortunately, due to the projection in (4.56) a centralized computation is required, which makes this approach impractical.

Partial Dual Decomposition. In the t-th iteration we update

$$x_s^{(t+1)} := \min_{x_s}\{U_s(x_s) - (\gamma^{(t)T}W_s)x_s\},\ s = 1, \ldots, S, \tag{4.57}$$

$$\gamma^{(t+1)} := \left[\gamma^{(t)} - \alpha(y - Wx^{(t+1)})\right]^+. \tag{4.58}$$

$$y^{(t+1)} = \arg\max_{y \geq 0}\{\gamma^{(t)T}y|Hy \preceq c\} \tag{4.59}$$

Subproblem (4.58) is solved in a centralized way, making this approach also inconvenient.

Chapter 5

Performance for the Quantized Method

In the previous chapters, we saw many algorithms that need message passing between the nodes and the links. In this chapter, we examine the scenario where the nodes of the network communicate with each other, passing useful information iteratively. Thus, in our case, the numbers they exchange are not continuous-valued information (real numbers), but instead quantized information. The problem of communication between agents of a network, to solve an optimization problem, has gained much attention in networking literature [27],[28],[29],[30]. Even though the communication in networks is not perfect, but affected by some kind of noise, which can be either a random additive noise or produced by a quantization, node failures and delays, we will focus on the convergence of the algorithm ignoring all the other problems that may occur. More specifically, we discuss a "quantized" extension of the subgradient method in the basic NUM.

5.1 Quantized basic NUM

In the basic NUM (3.12) it is possible not to use any explicit message passing since we can measure $\lambda^{(s)}$ by each source s as the queuing delay and each link can measure the aggregate rate by the total traffic load. But if we want to set up the network before it goes into usage, then we have to apply the algorithm with message passing. So, we quantize only the variables that must be communicated between the master problem and the subproblems.

So, we solve for each Source this subproblem for a fixed λ^Q starting by positive $\lambda^Q = 1_l$.

$$\underset{x_s \geq 0}{\text{maximize}} \quad cost_s\sqrt{x_s} - \lambda_s^{Q(t)} x_s, \quad \forall s \tag{5.1}$$

where $\lambda_s^{Q(t)}$ is the quantized aggregate Lagrange multiplier λ_s for all links that x_s passes throw after t iterations. Note that on the calculation of the utility function at each source,

we do not have to use quantized x because it is information that the source has. Now on the Master problem update λ^Q following the subgradient method we have

$$\lambda_l^Q(t+1) = \left[\lambda_l^Q(t) - \alpha\left(c_l - (Ax^Q)_s\right)\right]^+. \tag{5.2}$$

5.2 Numerical Example

We run the algorithm with $L = 7$ links and $s = 5$ sources with random capacity of links and with utility function $U_s(x_s) = cost_s\sqrt{x_s}$. We used the command "fi" on Matlab with word size $16, 12, 10$ and 8 bits. We used best-precision fraction length because the size of the message is the one we want to test. If the values of the quantized variables where bigger then we would lose precision on decimals. We can see some interesting results in Figures 5.2 and 5.3. The algorithm behaves normally at the start but it loses precision while it approaches the optimal value. If we are pleased with an error $|U(x^{(t)}) - U^*|$ at 10^{-2} and we do not want to use more than 10 bits for message passing then we can choose the trade off. If we need better precision, then we need to use more bits.

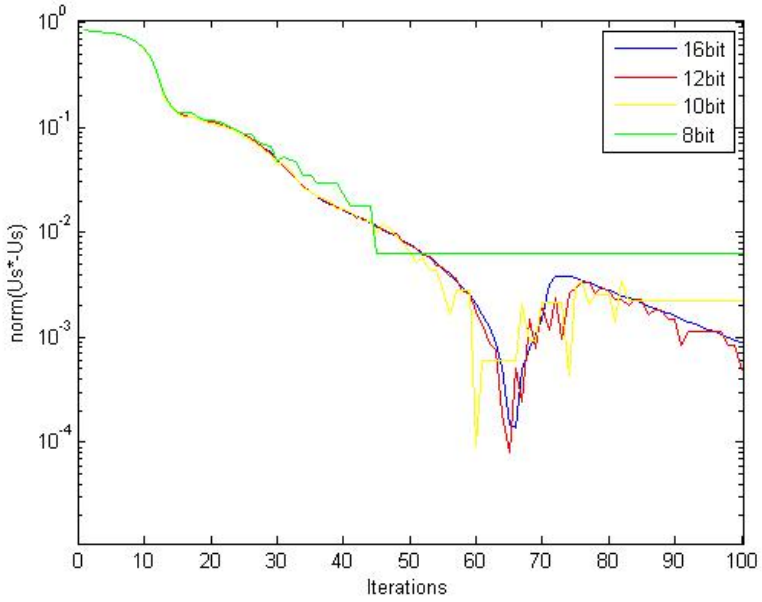

Figure 5.1: The performance of the quantized dual decomposition algorithm.

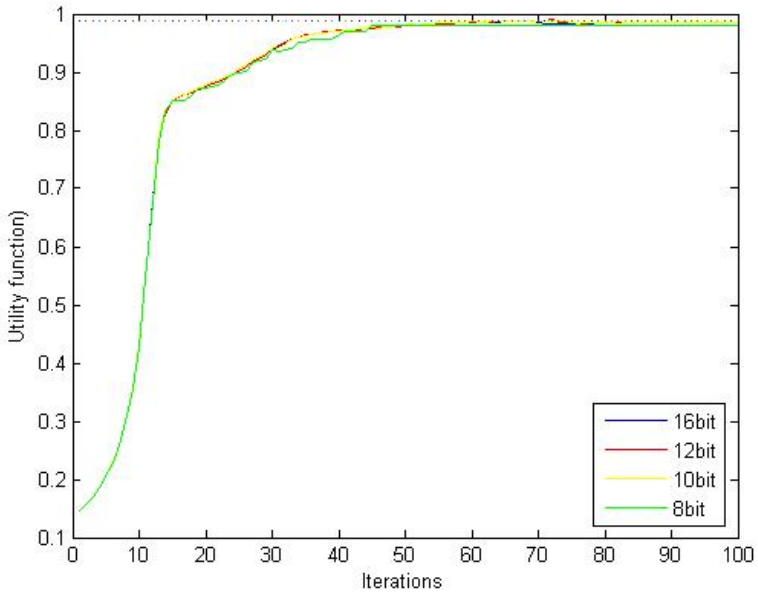

Figure 5.2: The utility function convergence for the different word size of the transmitted message.

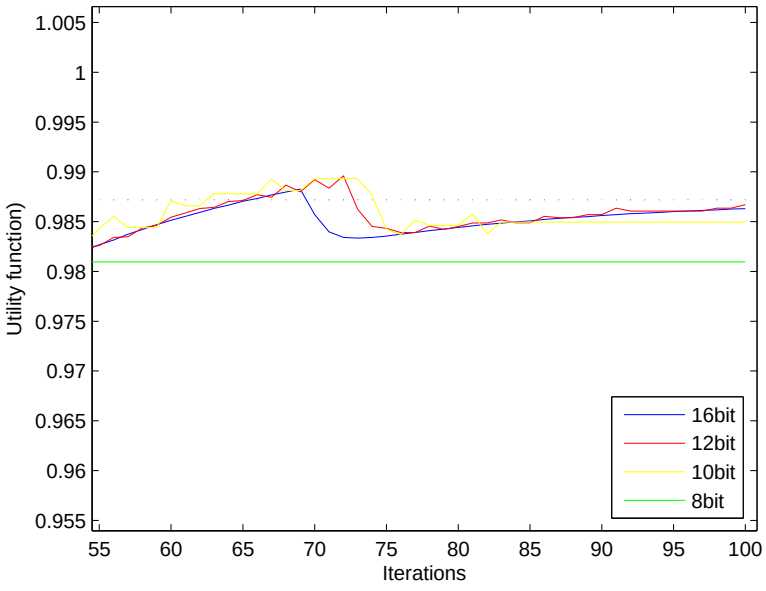

Figure 5.3: One closer look at Figure 5.2, which we can see the difference of the quantization.

Chapter 6

Discussion and Future Work

In this thesis, we have considered two basic and nine composite optimization algorithms that only rely on peer-to-peer communication and are suitable for use in networked systems. Furthermore, we have presented the solutions of six problems involving center-free resource allocation. We have also discussed the basic convex concepts and the use of the subgradient and subgradient methods. Finally, we have implemented and evaluated the Dual decomposition algorithm for the basic NUM problem with quantized message passing. In this chapter, we summarize and discuss the work of the thesis. In addition, we also outline some future work.

We start with an important question.

6.1 How Should NUM Problems be Posed, Decomposed, and Solved?

The answer to that question will of course very much depend on the specifics of the problem. However, looking at the literature, it is possible to see some patterns. In [31] we can see the effort to categorize some of the existing approaches to solving NUM problems in the following three blocks:

Problem Formulation The problem formulation is of paramount importance. The model has to capture all relevant phenomena while still being sufficiently simple; this is the classic fidelity versus tractability tradeoff. In addition, the difference between a good problem formulation and a bad one can be the difference that makes it solvable or not. Specifically, convex and decomposable optimization problems are desired. We can transform the problem formulation in order to make it simpler. We can change the variables or add new ones like the auxiliary variable at LP example at Chapter 3. In some cases, we may have to combine variable and constraint transformations to simplify our problem.

Decomposition In order to engineer a decentralized algorithm, the optimization problem has to be split into several subproblems that can be solved in a distributed way using some coordination scheme. Most often, either a dual or primal decomposition technique is chosen; see Chapter 3. The details are shown in Figure (6.1).

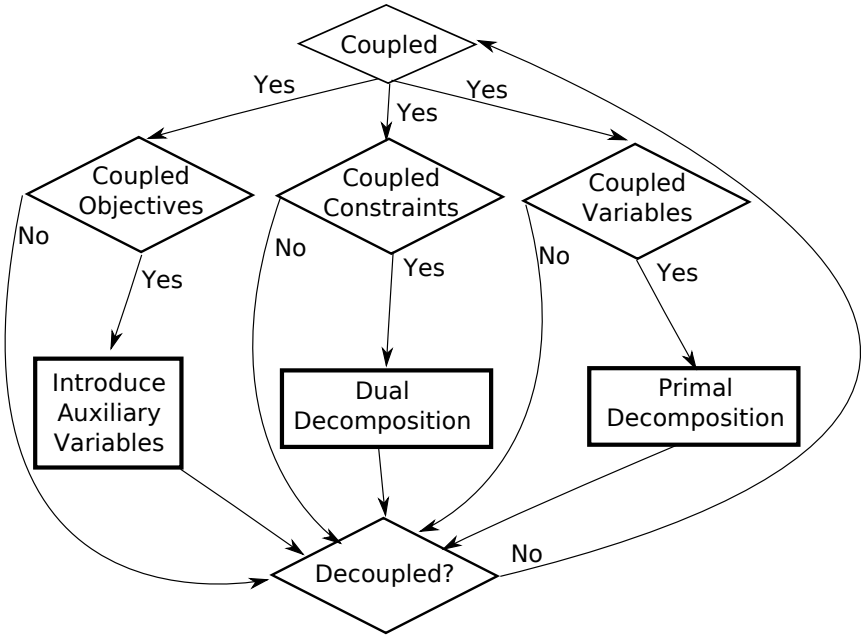

Figure 6.1: A decision chart for the use of decomposition methods.

To capture all decomposition approaches previously pursued in the literature, the flowchart would have to be huge, with the effect that it would be practically useless. Therefore, we are forced to compromise, and the flowchart we present captures, in our opinion, the most crucial steps and algorithmic approaches in solving NUM problems. The aim is to find patterns in the existing literature, to make it easier to get an overview what has been done, and to visualize the inherent steps in the NUM framework.

Optimization Techniques The resulting subproblems have to be solved in some way; numerous methods can of course be used, some of the most common methods are (sub)gradient, newton, cutting plane, interior point, other heuristics methods or other ascent method.

6.2 Synchronous vs. Asynchronous algorithms?

The subproblems can either be solved in a synchronous or an asynchronous fashion. The synchronous case is much simpler to analyse, whereas the asynchronous case is more general. Maintaining synchronous operations in a large network is demanding, and most real protocols will run asynchronously. Therefore, it is comforting if the analysis also guarantees proper operation of the algorithm under asynchronous operation as well. We believe that the algorithms presented in this thesis work well in both cases, the performance difference depends on the problem and must be investigated further.

6.3 Future work

We have tested many NUM algorithms for a fixed network and a fixed incidence matrix. What if we try to test the stability of these methods on a network that dynamically changes. Would the methods be stable or not?

In our networks, we knew or we could calculate the link capacities. What if we had to estimate them in order to solve the problem. Would that be a problem to work with estimates of the capacities?

We used the subgradient method for all our algorithms. What if we try to test other algorithms, maybe suboptimal, for the calculation of our variables?

We tried to quantize the message passing information in order the algorithm works properly. Many networks have message failures and noisy transmission that may affect the algorithm with a wrong value. How can be protected by something like that?

Bibliography

[1] Daniel P. Palomar, Mung Chiang. *"A Tutorial on Decomposition Methods For Network Utility Maximization",* IEEE J.Sel. Area Commun., vol.24,no 8, pp 1439-1451, Aug. 2006

[2] Daniel P. Palomar, Mung Chiang. *"Alternative distributed algorithms for network utility maximization: Framework and applications",* IEEE J.Automatic Control, IEEE Transactions on, vol.52,no 12,pp 2254-2269, 2007

[3] G. B. Dantzig and P. Wolfe. *"Decomposition principle for linear programs".* Operations Research, 8:101-111,1960

[4] Ford, L. R., and Delbert Ray Fulkerson. *"Flows in networks".* Vol. 1962. Princeton University Press: Princeton, 1962. APA

[5] Lasdon, Leon S. *"Optimization theory for large systems".* Courier Dover Publications, 1970.

[6] D. P. Bertsekas. *"Nonlinear Programming".* Athena Scientic, second edition, 1999

[7] F. Kelly, A. Maulloo, and D. Tan. *"Rate control for communication networks: Shadow prices, proportional fairness and stability".* Journal of the Operational Research Society, vol.49, no 3, pp 237-252, March 1998.

[8] Srinivas Shakkottai, Rayadurgam Srikant, *"Network optimization and control".* Now Publishers Inc, 2008

[9] Palomar, Daniel P, Eldar, Yonina C, *"Convex optimization in signal processing and communications".* 2010, Cambridge university press.

[10] R. Srikant and L. Ying, *"Communication Networks: An Optimization, Control and Stochastic Networks Perspective".* New York: Cambridge Univ Pr, 2014.

[11] X. Lin, N. Shroff, and R. Srikant, *"Tutorial on cross-layer optimization in wireless networks"*, IEEE J. Select. Areas Commun., vol. 24, no. 8, pp. 1452-1463, Aug. 2006.

[12] Z.-Q. Luo and W. Yu *"An introduction to convex optimization for communications and signal processing,"* IEEE J. Select. Areas Commun., vol. 24, no. 8, pp.1426-1438 2006

[13] S. Boyd and L. Vandenberghe. *"Convex Optimization."* Cambridge University Press, 2004.

[14] S. Boyd and L. Vandenberghe, Subgradient notes, Stanford University, http://stanford.edu/class/ee364b/lectures/subgradients_notes.pdf

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *"Distributed optimization and statistical learning via the alternating direction method of multipliers."* Foundations and Trends in Machine Learning, 3(1):1124, 2011.

[16] Problem description. http://www.stanford.edu/class/ee392o/decomposition_example.pdf

[17] M. Chiang, *"Balancing transport and physical layer in wireless multihop networks: Jointly optimal congestion control and power control,"* IEEE J. Sel. Areas Commun., vol. 23, no. 1, pp. 104-116, Jan. 2005.

[18] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, *"Layering as optimization decomposition: A mathematical theory of network architectures,"* Proc. IEEE, vol. 95, no. 1, pp. 255-213, Jan. 2007

[19] S. H. Low, *"A duality model of TCP and queue management algorithms,"* IEEE/ACM Trans. Netw., vol. 11, no. 4, pp. 525-536, Aug.2003.

[20] B. Johansson and M. Johansson, *"Primal and dual approaches to distributed cross-layer optimization,"* presented at the 16th IFAC World Congr., Prague, Czech Republic, 2005.

[21] L. Xiao, M. Johansson, and S. Boyd, *"Simultaneous routing and resource allocation via dual decomposition,"* IEEE Trans. Commun., vol. 52, no. 7, pp. 1136-1144, Jul. 2004.

[22] M. H. D. Katabi and C. Rohrs, *"Internet congestion control for high bandwidth-delay product networks,"* presented at the ACM Sigcomm, Aug. 2002.

[23] N. Dukkipati, M. Kobayashi, Z. S. Rui, and N. McKeown, *"Processor sharing flows in the internet,"* presented at the 13th Int. Workshop Quality of Service (IWQoS), Passau, Germany, Jun. 2005.

[24] X. Lin and N. B. Shroff. *"Utility maximization for communication networks with multipath routing,"* IEEE Trans. Autom. Control, vol. 51, no. 5, pp. 766-781, May 2006.

[25] J. Wang, L. Li, S. H. Low, and J. C. Doyle, *"Cross-layer optimization in TCP/IP networks,"* IEEE/ACM Trans. Netw., vol. 13, pp. 582-595, Jun. 2005

[26] L. Chen, S. H. Low, and J. C. Doyle, *"Joint congestion control and media access control design for ad hoc wireless networks"*, presented at the IEEE INFOCOM, Miami, FL, Mar. 13-27, 2005.

[27] A. Kashyap, T. Basar, and R. Srikant, *"Quantized consensus,"* Automatica, vol. 43, no. 7, pp. 1192-203, 2007.

[28] R. Carli, F. Fagnani, P. Frasca, T. Taylor, and S. Zampieri, *"Average consensus on networks with transmission noise or quantization,"* in Proceedings of the European Control Conference, 2007.

[29] R. Carli, F. Fagnani, A. Speranzon, and S. Zampieri, *"Communication constraints in coordinated consensus problem,"* in Proceedings of the IEEE American Control Conference, 2006, pp. 4189-94.

[30] A. Nedic, A. Olshevsky, A. Ozdaglar, and J.N. Tsitsiklis, *"Distributed subgradient methods and quantization effects,"* presented at the Proceedings of IEEE CDC, 2008

[31] B. Johansson, *"On distributed optimization in networked systems,"* Ph.D. dissertation, Royal Institute of Technology (KTH), Dec. 2008, tRITA-EE 2008:065.