Abstract

We review and analyze nonlinear programming approaches to modeling and solving certain flow problems in telecommunications, transportation and supply chain management. We emphasize the common aspects of telecommunications and road networks, and indicate the importance of game theoretic and equilibrium approaches. Algorithms based on the Frank-Wolfe method are developed in depth and their implementations on sequential and parallel machines are discussed and evaluated for large-scale real-world networks. Several research directions are also stated.

Keywords: nonlinear optimization, Frank-Wolfe algorithm, conditional gradient algorithm, flow deviation algorithm, regularized Frank-Wolfe algorithm, simplicial decomposition, routing, flow assignment, traffic assignment, transportation problems, equilibrium flows, team games, Nash equilibrium, Wardrop’s principle, bilevel programming, Stackelberg game, capacity assignment, network design, supply chain management, parallel computing
1 Introduction

Traffic planning and transportation analysis have been subject to intensive quantitative research during the last 40 years which has resulted into a rich bibliography concerning both modeling aspects and development of solution algorithms (see e.g. [58, 115, 135, 152, 123]). The purpose of this research is to assist planners in predicting flow pattern fluctuations during a day as well as flow pattern changes which may occur as result of decisions that affect the network topology, lane capacity, traffic lights, tolls and other traffic control mechanisms. The cornerstone of such analysis are descriptive traffic equilibrium models which allow planners to perform the necessary analysis with the assistance of computer simulations instead of physical experiments. These models are usually referred to as traffic assignment models and have an important role in evaluating the performance of a traffic network with respect to congestion, safety, user satisfaction, etc.

Telecommunications have always been the subject of application for advanced mathematical techniques such as queuing theory (e.g. Kleinrock [93, 94]). With the advent of computer networks, research on and applications of optimization modeling and algorithms to problems in the area of data communication networks intensified in 70’s (e.g. Schwartz [148]). Following the rapid technological development in telecommunications and the derivation of new efficient optimization methods, research and applications of optimization techniques to communications kept an increasing pace in the 80’s (e.g. Seidler [149]) and 90’s (e.g. Bertsekas and Gallager [21]) culminating the recent years with the new opportunities offered by the explosive expansion of fiber, broadband, mobile and wireless networking. Most analysis or design problems in communication systems can be stated in terms of optimizing, i.e., minimizing or maximizing, some performance or utility function in several variables the values of which must satisfy a set of prespecified requirements. These optimization problems quite often turn to be of combinatorial nature, and thus belong to the sphere of combinatorial optimization, however, nonlinear optimization has also been successfully applied to several problems arisen in the context of communication systems, including the areas of wireless networks, high speed Internet, equalization to broadband access, network topology, routing, etc. see e.g. the handbook [143].

Supply chain management concerns the management of material and information flows between facilities, such as vendors, manufacturing, and assembly plants, and distribution centers. In general there are three major
stages in the supply chain: procurement, production and distribution, each of which may concern several facilities in different locations. An important class of decision problems in a supply chain considers the case where a firm with several plants must supply several demand points. In these problems the objective is to minimize the total cost consisting of production costs including costs of building, maintaining and operating facilities and/or distribution-transportation costs including the cost of shipping production quantities from facilities to warehouses or to retailers. Depending on the assumption about the structure of production cost and the nature of the demand faced by the firm the production-distribution link can take on many different forms which lead often to complex, large-scale models. Therefore, the use of scientific approaches is necessary in order to certify a good performance of the supply chain or to detect opportunities for improving it. Operations Research has proven to be a powerful tool in this respect. Geoffrion and Powers [71] summarize some of the main reasons for the increasing role of optimization techniques in the design of distribution systems. In these notes we are concerned only with problems dealing with the minimization of differentiable convex objective function subject to a set of linear constraints which cover a wide range of specific problems in the supply chain, namely, the stochastic transportation problem, the transportation problem with costumer competition for the offered service level and the production transportation problem where the demand of the customers is satisfied by the presence of an intermediary.

The general form of the problems studied in nonlinear optimization may be stated as follows:

\[(NLP) \quad \min_{x \in S} f(x) \quad (1)\]
\[\text{s.t.} \quad g(x) \leq 0 \quad (2)\]
\[h(x) = 0 \quad (3)\]
\[x \in S, \quad (4)\]

where \(S \subseteq \mathbb{R}^n\), \(f : x \in S \to f(x) \in \mathbb{R}\), \(g : x \in S \to g(x) \in \mathbb{R}^k\), and \(h : x \in S \to h(x) \in \mathbb{R}^m\). The function \(f(\cdot)\) is the objective function of the problem, (2)-(4) are the problem constraints which define the feasible region of the problem, i.e., the set of points \(x \in \mathbb{R}^n\) that are admissible candidate solutions to \(NLP\). We assume that (4) models simple constraints such as bounds on the values of \(x\). Problem \(NLP\) is a nonlinear optimization problem as long as at least one of the involved functions is nonlinear.
In the present lecture notes we are concerned with classical nonlinear programming approaches to solving problems of type NLP. This means in particular that the methods and algorithms discussed in Section 4 are able to attack only special instances of NLP, requiring, for example, continuity and differentiability of the involved functions in order to produce local minima (c.f. Definition 1) and certain convexity (c.f. Definitions 3 and 10) conditions to be satisfied in order to guarantee global optimality (c.f. Definition 1) of the produced solution. Hence some important cases of NLP, such as mixed integer nonlinear optimization problems, stochastic optimization problems, and global optimization problems, will not be discussed. Thus, certain very interesting problems in telecommunications, including hub or switching network design (see e.g. [75, 82, 131]), traffic management, capacity expansion and design of ATM networks (e.g. [22, 26, 30, 51]), pricing and costs in e-services (e.g. [100]), e.t.c. will not be discussed in the present lecture notes. Here we place the emphasis on routing (or flow assignment) and capacity assignment problems for several reasons: (a) similar problems have been studied for road traffic networks for several decades, however, the crossover between the two fields has been minimal, (b) the corresponding road traffic network problems have been based on game theoretic concepts since the seminal works of Wardrop [157] and Dafermos [44], while only recently, due to the deregulation of telecommunications and the increased competition in Internet services, has this game theoretic approach gained the attention of the telecommunications community, showing however impressive activity [2], (c) although we are basically concerned with networks of fixed topology, there are lessons to be learned from the findings of several decades and bring them into the study of networks with dynamically changing topology, and (d) several important improvements on algorithms for road traffic flow assignments can be carried over to communications.

The exposition of the lecture notes is as follows: Section 2 is dedicated to a brief overview of the basic theory of nonlinear optimization. In Section 3, several classical problems of routing and capacity assignment, as well road traffic flow assignment and network design are presented and emphasis is based on the interrelations of the two areas. The derivation of the models are based on the concept of team games, and distinction is made between, on one hand, equilibrium or user optimum flows and, on the other hand, system optimum flows. Several other nonlinear optimization problems, such as energy conserving routing in wireless ad-hoc networks and the extension of lifetime of power constrained networks, are also briefly surveyed. In Section 4,
nonlinear programming algorithms able to solve problems stated in Section 3 are discussed, several of them in depth. Emphasis is placed on the Frank-Wolfe algorithm [57], known also as the flow deviation method [59] to the communications community, as several new methods have been proposed in recent years which improve the performance of the original algorithm. Other methods, not related to Frank-Wolfe are also surveyed. We conclude with Section 5 where several future research issues are presented.

2 Brief Overview of the Basic Theory

In this section we briefly review a few basic concepts from the theory of nonlinear optimization needed in subsequent sections. There are several excellent books (e.g. [6, 17, 20, 116]) on the subject and the reader should turn to them for further details. The few concepts of game theory needed are borrowed from [126, 123, 125]. The presentation will be based partly on NLP and partly on the following statement of a nonlinear optimization problem:

\[ \min_{x \in \mathcal{X}} f(x) \]

where \( \mathcal{X} \subseteq \mathbb{R}^n \) is the feasible region, and \( f(\cdot) \) is the objective function of the problem. A point \( x \in \mathbb{R}^n \) is called feasible point to \( P \) if \( x \in \mathcal{X} \), otherwise \( x \) is infeasible. There are two solution concepts associated with any nonlinear programming problem and these are introduced next:

**Definition 1** A point \( x^* \) is a local minimum point of \( P \) if

1. \( x^* \in \mathcal{X} \), i.e., it is feasible, and
2. there exists a neighborhood \( \mathcal{N}_\epsilon(x^*) = \{ x : \|x - x^*\| \leq \epsilon \} \), \( \epsilon > 0 \), such that \( f(x^*) \leq f(x) \), \( \forall x \in \mathcal{N}_\epsilon(x^*) \cap \mathcal{X} \).

A point \( x^* \) is a global minimum point of \( P \) if

1. \( x^* \in \mathcal{X} \), i.e., it is feasible, and
2. \( f(x^*) \leq f(x) \), \( \forall x \in \mathcal{X} \).
The point $x^*$ is often referred to as a local (or global) optimal point or optimal solution.

A global minimum is always a local minimum. However, the reverse of this statement is true only for very specific instances of $P$, see for example Theorem 2. Moreover, a nonlinear optimization problem may have no optimal solution unless it satisfies certain conditions, such as lower semi-continuity of the objective function and compactness of the feasible set.

**Definition 2** The function $f(x)$ defined on $X \subseteq \mathbb{R}^n$ is said to be lower semicontinuous in $X$ if the sets $L(b) = \{x \in X| f(x) \leq b\}$ are closed relative to $X$ for any $b \in \mathbb{R}$.

A continuous function is also lower semicontinuous. The reverse of the statement is not true. Existence of a solution to $P$ is guaranteed by the next theorem.

**Theorem 1** If $X$ is a nonempty compact set and the function $f(x)$ is lower semicontinuous in $X$, then $P$ has a solution $x^* \in X$.

To guarantee the global optimality of the solution, additional requirements must be posed on $P$. Convexity of the objective function and the feasible region are two such requirements and are formally introduced next.

**Definition 3** The set $X \subset \mathbb{R}^n$ is convex if for all $x^1, x^2 \in X$ it satisfies the inclusion

$$\alpha x^1 + (1 - \alpha)x^2 \in X$$

for all $\alpha \in [0, 1]$, that is, the convex combination of any two points in $X$ is also in $X$. The function $f(x)$ is convex in the convex set $X$ if for all $x^1, x^2 \in X$ and $\alpha \in (0, 1)$ it satisfies the inequality

$$f(\alpha x^1 + (1 - \alpha)x^2) \leq \alpha f(x^1) + (1 - \alpha)f(x^2).$$

The function is strictly convex if strict inequality holds in the above relation. Moreover, the function $-f(x)$ is (strictly) concave if $f(x)$ is (strictly) convex.

The next theorem guarantees the global optimality of any local optimum in $P$ under convexity assumptions.
Theorem 2 \textit{If $\mathcal{X}$ and $f(x)$ are convex in $\mathbf{P}$, then any local minimum point is also global. Moreover, if $f(x)$ is strictly convex the minimum point is unique.}

The case when the feasible region of a nonlinear program is defined by linear inequalities is of particular interest. The next few definitions and theorems address this issue.

\textbf{Definition 4} \textit{A polyhedron $\mathcal{X}$ in $\mathbb{R}^n$ is the intersection of a finite number of closed halfspaces, i.e., $\mathcal{X} = \{x \in \mathbb{R}^n | a_i^T x \leq b_i, i = 1, \ldots, m\}$, where $a_i \in \mathbb{R}^n$ are constant vectors and $b_i$ are real numbers. A bounded polyhedron is called a polytope.}

The convex hull of a finite number of points is the collection of all convex combinations of these points. Such a convex hull is always a polytope. The simplest example of a polytope is the simplex. Polyhedra and polytopes are convex sets of special structure, in particular they possess certain finite characteristics.

\textbf{Definition 5} \textit{The point $x$ is an extreme point or a vertex of the polyhedron $\mathcal{X} \subset \mathbb{R}^n$ if $x = \alpha x^1 + (1-\alpha)x^2$ with $x^1, x^2 \in \mathcal{X}$ and $\alpha \in (0, 1)$ implies that $x = x^1 = x^2$. That is, an extreme point cannot be written as a convex combination of two distinct points in $\mathcal{X}$.}

\textbf{Definition 6} \textit{The nonzero vector $d \in \mathbb{R}^n$ is called a direction of the polyhedron $\mathcal{X}$ if for every $x \in \mathcal{X}$, $x + \alpha d \in \mathcal{X}$ for all $\alpha \geq 0$. The direction $d$ of $\mathcal{X}$ is an extreme direction of $\mathcal{X}$ if $d = \alpha d^1 + \beta d^2$ for $\alpha, \beta > 0$ implies $d^1 = \gamma d^2$ for some $\gamma > 0$. That is, an extreme direction cannot be written as a positive linear combination of two distinct directions in $\mathcal{X}$.}

\textbf{Theorem 3} \textit{The number of extreme points or vertices of a polyhedron is finite. The number of extreme directions of a polyhedron is finite.}

Caratheodory’s theorem states that a point in the convex hull of a set can be represented as a convex combination of a finite number of points in the set. The theorem in the case of polyhedral sets is also known as the theorem of inner representation and states that any point in a polytope can be expressed as a convex combination of the extreme point of the polytope and that any point in a polyhedron can be expressed as a convex combination of its extreme points plus a linear combination of its extreme directions.
Theorem 4 A polytope is the convex hull of its vertices, that is, if \( \bar{x}^1, \bar{x}^2, \ldots, \bar{x}^K \) are the extreme points of a polytope \( \mathcal{X} \subset \mathbb{R}^n \), then

\[
\mathcal{X} = \left\{ x \in \mathbb{R}^n \mid x = \sum_{k=1}^{K} \alpha_k \bar{x}^k, \sum_{k=1}^{K} \alpha_k = 1, \alpha_k \geq 0 \text{ for } k = 1, \ldots, K \right\}
\]

In particular, since \( \mathcal{X} \subset \mathbb{R}^n \), then at most \( n + 1 \) extreme points are needed in order to represent any point \( x \in \mathcal{X} \).

Moreover, if \( \mathcal{X} \subset \mathbb{R}^n \) is an unbounded polyhedron with \( K \) extreme points \( \bar{x}^1, \bar{x}^2, \ldots, \bar{x}^K \) and \( L \) extreme directions \( \bar{d}^1, \bar{d}^2, \ldots, \bar{d}^L \), then \( x \in \mathcal{X} \) if and only if

\[
x = \sum_{k=1}^{K} \alpha_k \bar{x}^k + \sum_{\ell=1}^{L} \beta_\ell \bar{d}^\ell
\]

\[
\sum_{k=1}^{K} \alpha_k = 1
\]

\[
\alpha_k \geq 0, \forall k
\]

\[
\beta_\ell \geq 0, \forall \ell
\]

Several algorithms for the solution of \( \mathbf{P} \) are based on the notion of feasible directions of descent. That is, given a point \( x^k \in \mathcal{X} \), a new point \( x^{k+1} \in \mathcal{X} \) is produced such that \( f(x^{k+1}) < f(x^k) \), where \( x^{k+1} = x^k + \alpha_k d^k \) for some \( \alpha_k > 0 \) and some vector \( d^k \neq 0 \), called a direction. Thus, there is a particular interest in the behavior of the objective function \( f(\cdot) \) along such directions. The concept of directional derivative provides in general the necessary information.

Definition 7 Let \( f(x) \) be defined on the nonempty set \( \mathcal{X} \subset \mathbb{R}^n \) and suppose that \( \bar{x} + \alpha d \in \mathcal{X} \) for \( \bar{x} \in \mathcal{X} \), \( d \neq 0 \) a vector in \( \mathbb{R}^n \), and \( \alpha > 0 \) sufficiently small. The directional derivative of \( f(x) \) at \( \bar{x} \) in the direction \( d \) is given by the following limit if it exists:

\[
\lim_{\alpha \to 0^+} \frac{f(\bar{x} + \alpha d) - f(\bar{x})}{\alpha}
\]
The limit in the previous definition with the values $-\infty$ and $\infty$ permitted exists for convex, and consequently for concave, functions.

**Theorem 5** Let $\mathcal{X}$ be nonempty and convex set and assume that $f(x)$ is convex, then the limit in the definition of the directional derivative exists.

If the gradient of the objective function exists, the directional derivative can be calculated as the scalar product of the gradient and the direction.

**Theorem 6** If $f(x)$ has the gradient $\nabla f(\bar{x})$ at $\bar{x} \in \mathbb{R}^n$ and $d \neq 0$ is a vector in $\mathbb{R}^n$, then the directional derivative of $f(x)$ at $\bar{x}$ in the direction $d$ is $\nabla f(\bar{x})^T d$.

Whenever this scalar product is negative, the direction is descent, that is, a small step along it will decrease the current function value.

**Definition 8** Let $f(x)$ have the gradient $\nabla f(\bar{x})$ at $\bar{x}$. Then the vector $d \neq 0$ is descent direction of $f(x)$ at $\bar{x}$ if $\nabla f(\bar{x})^T d < 0$. The set $D(\bar{x}) = \{d \in \mathbb{R}^n | \nabla f(\bar{x})^T d < 0 \}$ is the set of descent directions of $f(\cdot)$ at $\bar{x}$.

However, not all descent directions at a point are admissible.

**Definition 9** Let $\bar{x}$ be feasible to $P$. Then the vector $d \neq 0$ is a feasible direction from $\bar{x}$ if there exists a $\bar{\alpha} > 0$ such that $\bar{x} + \alpha d$ is feasible to $P$ for all $\alpha$ satisfying $0 \leq \alpha \leq \bar{\alpha}$. The set of all feasible directions from $\bar{x}$ will be denoted by $F(\bar{x})$.

Clearly, a point $\bar{x}$ feasible to $P$ for which $F(\bar{x}) \cap D(\bar{x}) \neq \emptyset$ cannot be a (local) minimum point to $P$. Indeed, the next theorem states that if $F(\bar{x}) \cap D(\bar{x}) \neq \emptyset$, then it is always possible to find another feasible point with lower objective function value.

**Theorem 7** If $f(x)$ is differentiable at $\bar{x}$, then it is continuous and has the gradient $\nabla f(\bar{x})$ at $\bar{x}$. Moreover, if $f(x)$ is differentiable on the segment joining $x^1$ and $x^2$, then the function $\phi(\alpha) = f(\alpha x^1 + (1-\alpha)x^2)$, defined for $\alpha \in [0, 1]$, is differentiable in $[0, 1]$ and has the derivative $\phi'(\alpha) = (x^1 - x^2)^T \nabla f(\alpha x^1 + (1-\alpha)x^2)$, that is, $\phi'(\alpha)$ is the directional derivative of $f(x)$ at $\alpha x^1 + (1-\alpha)x^2$ in the direction $x^1 - x^2$. If $\phi'(0) < 0$, i.e., $(x^1 - x^2)^T \nabla f(x^2) < 0$, then there is a $\alpha \in (0, 1)$ such that $\phi(\alpha) < \phi(0)$, or equivalently $f(\alpha x^1 + (1-\alpha)x^2) < f(x^2)$, for all $\alpha \in (0, \bar{\alpha})$. 

9
Differentiable convex functions have two important properties in terms of their gradient; their linearization by the tangent hyperplane at a given point provides an underestimation as stated by Theorem 8 below, and there exist necessary and sufficient conditions for an optimal solution as stated by Theorem 9 in terms of variational inequalities.

**Theorem 8**  Let \( f(x) \) be a differentiable function on a nonempty open set \( X \subseteq \mathbb{R}^n \). Then \( f(x) \) is convex if and only if for any \( y \in X \) the inequality
\[
f(x) \geq f(y) + \nabla f(y)^T(x - y), \quad \forall x \in X
\]
holds.

**Theorem 9**  Let \( f(x) \) be a differential convex function defined on the convex set \( X \). Then \( x^* \) is a global optimal solution to \( P \) if and only if
\[
\nabla f(x^*)^T(x - x^*) \geq 0, \quad \forall x \in X.
\]
Moreover, if \( X \) is open then \( x^* \) is an optimal solution if and only if \( \nabla f(x^*) = 0 \).

The properties of the above theorem are not shared only by convex functions but also by pseudoconvex functions, which are introduced in the next definition.

**Definition 10**  The differentiable function \( f(x) \) defined on a nonempty open set \( X \subseteq \mathbb{R}^n \) is said to be pseudoconvex if
\[
\nabla f(x^1)^T(x^2 - x^1) \geq 0 \Rightarrow f(x^2) \geq f(x^1), \quad \text{or equivalently,}
\]
\[
f(x^2) < f(x^1) \Rightarrow \nabla f(x^1)^T(x^2 - x^1) < 0,
\]
for all \( x^1, x^2 \in X \). Moreover, the function \( -f(\cdot) \) is then called pseudoconcave.

In the absence of (pseudo-) convexity, global optimality of the local minimum point \( x^* \) cannot be guaranteed in \( P \). In particular, if \( f(x) \) in \( P \) is a differentiable concave function we have the following results:

**Theorem 10**  Let \( f(x) \) be a differential concave function defined on the convex set \( X \). If \( \bar{x} \) is a local optimal solution to \( P \) then
\[
\nabla f(\bar{x})^T(x - \bar{x}) \geq 0, \quad \forall x \in X.
\]
(5)

Moreover, if \( X \) is a nonempty polytope, then there exists an extreme point \( \bar{x} \in X \) which is an optimal solution of \( P \).
Actually, 5 in the above theorem does not require the concavity of the objective function. If in particular \( \nabla f(\bar{x})^T(x^k - \bar{x}) < 0 \) for some \( x^k \in \mathcal{X} \), then \( d^k = x^k - \bar{x} \) is a feasible direction of descent, and \( f(\cdot) \) can be decreased from its current value \( f(\bar{x}) \) by moving to a new feasible point \( x^{k+1} = \bar{x} + \alpha_k (x^k - \bar{x}) \) for some suitably chosen \( \alpha_k > 0 \). It should also be clear that if \( \mathcal{X} = \mathbb{R}^n \), then conditions \( 5 \) take the form of the well-known first order necessary conditions \( \nabla f(\bar{x}) = 0 \).

When \( \mathcal{X} \) is given explicitly, the optimality conditions derived above can be expressed in a more manageable form. Suppose that \( \mathcal{X} = \{ g_i(x) \leq 0, i = 1, \ldots, \ell \} \) in \( \mathbf{P} \). Let \( \bar{x} \in \mathcal{X} \) and define the sets \( \mathcal{I}(\bar{x}) = \{ i \mid g_i(\bar{x}) = 0 \} \) and \( \mathcal{D}^L(\bar{x}) = \{ d \in \mathbb{R}^n \mid d^T \nabla g_i(\bar{x}) \leq 0, \forall i \in \mathcal{I}(\bar{x}) \} \). The set \( \mathcal{I}(\bar{x}) \) is the index set of the binding or active constraints at \( \bar{x} \), while the set \( \mathcal{D}^L(\bar{x}) \) gives a linearized approximation of the feasible region in a neighborhood of the feasible point \( \bar{x} \).

**Definition 11** The Kuhn-Tucker constraint qualification holds at \( \bar{x} \) if \( \mathcal{D}(\bar{x}) = \mathcal{D}^L(\bar{x}) \).

In general it is almost impossible to verify the Kuhn-Tucker constraint qualification directly. Therefore several more or less checkable sufficient conditions for the Kuhn-Tucker constraint qualification have been proposed in the literature. See e.g. [6] for a hierarchy of such constraint qualifications. Clearly, if \( \mathcal{D}^L(\bar{x}) \cap \mathcal{F}(\bar{x}) \neq \emptyset \), then \( \bar{x} \) is not a (local) minimum point since there exists a feasible descent direction (c.f. Theorem 7) from \( \bar{x} \) in which \( f(\cdot) \) strictly decreases. The next theorem provides the Karush-Kuhn-Tucker (KKT) necessary conditions for (local) optimality of \( \bar{x} \).

**Theorem 11** In NLP, let \( \mathcal{S} \) be a nonempty set with at least one interior point, assume that \( f, g \) and \( h \) are continuously differentiable. If \( \bar{x} \) is a local minimum point and some constraint qualification holds at \( \bar{x} \), then there exist \( \lambda \in \mathbb{R}^\ell \) and \( \mu \in \mathbb{R}^m \) such that

\[
-\nabla f(\bar{x}) = \lambda^T g(\bar{x}) + \mu^T h(\bar{x}) \quad (6)
\]

\[
\lambda^T g(\bar{x}) = 0 \quad (7)
\]

\[
\lambda \geq 0 \quad (8)
\]
Under certain convexity assumptions, the KKT necessary conditions are also sufficient for optimality. The next theorem states the Karush-Kuhn-Tucker (KKT) sufficient conditions for NLP.

**Theorem 12** Assume that the conditions of Theorem 11 on \( S, g \) and \( h \) are valid. Assume further that \( f \) and the component functions \( g_i(\cdot) \) of \( g(\cdot) \) are pseudoconvex and that \( h \) is linear, that is \( h(x) = Ax - b \). Then, any feasible point \( \bar{x} \) in NLP which satisfies the KKT-conditions (6)-(8) is a global minimum point to NLP.

Both Theorem 11 and Theorem 12 can be stated under far more mild assumptions, see e.g. [6, 17, 151, 116]. However, the assumptions made are satisfied by the problems considered in the lecture notes.

Sufficient optimality conditions for NLP can also be stated in terms of the Lagrangian function. For the problem NLP, define the Lagrangian function

\[
L(x, \lambda, \mu) = f(x) + \lambda^T g(x) + \mu^T h(x)
\]  

for \( x \in S \) and \( \lambda \geq 0 \), and consider the Lagrangian subproblem

\[
\Theta(\lambda, \mu) = \min_{x \in S} f(x) + \lambda^T g(x) + \mu^T h(x),
\]

defined for \( \lambda \geq 0 \).

**Theorem 13** Let \( \bar{x} \) be a solution to (10) for a given \( (\bar{\lambda}, \bar{\mu}) \). If \( \bar{x} \) is a feasible point to NLP and satisfies the conditions

\[
f(\bar{x}) + \bar{\lambda}^T g(\bar{x}) + \bar{\mu}^T h(\bar{x}) = \Theta(\bar{\lambda}, \bar{\mu})
\]  

\[
\bar{\lambda}^T g(\bar{x}) = 0
\]  

\[
\bar{\lambda} \geq 0,
\]

then \( \bar{x} \) is an optimal solution to NLP.

Conditions (11)-(13) are not necessary since for an optimal solution \( \bar{x} \) to NLP there may not be, in general, any corresponding \( (\bar{\lambda}, \bar{\mu}) \) with \( \bar{\lambda} \geq 0 \) that satisfies (11)-(13). Theorem 11 and Theorem 12 establish conditions under which (11)-(13) are necessary and sufficient for the global optimality of \( \bar{x} \); clearly by requiring certain convexity and constraint qualification assumptions to be satisfied.
We close this section by introducing the notion of non-cooperative games and the concept of non-cooperative equilibria. Let $\mathcal{X}_i \subset \mathbb{R}^n$ for $i = 1, \ldots, n$ be compact convex sets and assume that the functions $f_i : \prod_{i=1}^n \mathcal{X}_i \rightarrow \mathbb{R}$ are (pseudo-) convex, where $x = [x_1, \ldots, x_i, \ldots, x_n]$, $x_i \in \mathcal{X}_i$ for $i = 1, \ldots, n$, and $\mathcal{X} = \prod_{i=1}^n \mathcal{X}_i = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_n$. Let $\mathcal{N} = \{1, 2, \ldots, n\}$ denote a set of players or agents, $x_i$ denote a strategy of player $i$, $\mathcal{X}_i$ the set of strategies of player $i$, and $f_i$ the loss function of player $i$. Then the triple $G = [\mathcal{N}, \{\mathcal{X}_i\}_{i \in \mathcal{N}}, \{f_i\}_{i \in \mathcal{N}}]$ defines a game. The vector $x = [x_1, \ldots, x_n]$ of strategies $x_i \in \mathcal{X}_i$ chosen by the players is a situation, and the set $\mathcal{X}$ is the set of feasible situations. A situation $x^*$ in a non-cooperative game $G$ is called admissible for the player $i$ if for any other strategy $x_i \in \mathcal{X}_i$ for this player we have $f_i(x^*) \leq f_i(x_i^* - x_i, x_i^* + x_i^*)$, where $x_i^*$ denotes the partial vector $[x_1, \ldots, x_{i-1}]$, and $x_i^*$ denotes the partial vector $[x_{i+1}, \ldots, x_n]$. A situation $x^*$ which is admissible for all players is called a Nash [129] equilibrium situation:

**Definition 12** The situation $x^*$ is a Nash equilibrium of the game $G = [\mathcal{N}, \{\mathcal{X}_i\}_{i \in \mathcal{N}}, \{f_i\}_{i \in \mathcal{N}}]$ if

$$f_i(x^*) = \min_{x_i \in \mathcal{X}_i} f_i(x_i^* - x_i, x_i^* + x_i^*), \forall i \in \mathcal{N}$$ (14)

The game $G$ is a team game if $f_i(\cdot) = f(\cdot), \forall i \in \mathcal{N}$, that is, if all players share the same loss function. In such a case, a Nash equilibrium of the game is obtainable by solving the following problem

$$(\text{CPP}) \quad \min f(x_1, x_2, \ldots, x_n)$$

s.t. $x_i \in \mathcal{X}_i, \forall i \in \mathcal{N}$,

which is an optimization problem defined over a Cartesian product of sets.

Consider next an additional player 0 with loss function $g : \mathcal{Y} \times \prod_{i=1}^n \mathcal{X}_i \rightarrow \mathbb{R}^n$ and suppose that the loss functions of the other players $i \in \mathcal{N}$ are $f_i : \mathcal{Y} \times \prod_{i=1}^n \mathcal{X}_i \rightarrow \mathbb{R}^n$. Suppose that player 0 announces her decision first and that she is committed to implement it once announced. Player 0 will be called a leader. The other players $i \in \mathcal{N}$ react to such an announced leader’s strategy by optimizing their losses $f_i$. These players are called followers. Clearly, the leader must anticipate the reaction of the followers before she
announces her decision in order to optimize her loss \( g \). The situation can be stated as the following problem:

\[
(\text{GBP}) \quad \min g(y, x^*) \quad \text{subject to} \quad y \in \mathcal{Y}, \quad (15)
\]

where \( x^* \) is such that

\[
f_i(y, x^*) = \min_{x_{i-}, x_{i+}} f_i(y, x_{i-}, x, x_{i+}), \quad \forall i \in \mathcal{N}. \quad (17)
\]

If the followers are involved in a team game, then the problem can be rewritten as

\[
(\text{BP}) \quad \min g(y, x^*) \quad \text{subject to} \quad y \in \mathcal{Y}, \quad (18)
\]

where \( x^* \) is such that

\[
f(y, x^*) = \min_{x \in \mathcal{X}} f(y, x). \quad (20)
\]

Problems GBP and BP are called bilevel or two-level problems and they model a situation of a non-cooperative game known as the Stackelberg game. The leader’s problem (15)-(16) and (18)-(19) respectively, known as the first level problem, is the same in both cases. The second level problems (17) and (20) differ in the sense that in the first case the followers are involved in a Nash non-cooperative game, while in the second case the followers are involved in a team game. Clearly, in BP the modeled situation would have been the same if there was only one follower. Under certain conditions, both GBP and BP can be stated in an equivalent NLP or P form by replacing the second level problems by their corresponding optimality conditions either in the form of variational inequalities (c.f. inequality (5)) or in the form of Karush-Kuhn-Tucker (c.f. Theorem 11 and Theorem 12).

### 3 Models

#### 3.1 Traffic Assignment and Routing

Traffic equilibria models are descriptive in the sense that their aim is to predict flow patterns and travel times which are the results of the network users’ choices with regard to routes from their origins to their destinations. The input to the model is a complete description of the proposed or existing...
transportation system. The models are based on the behavioral assumption that “the journey times on all the routes used are equal, and less than those which would be experienced by a single vehicle on any unused route” [157]. This is Wardrop’s first condition, also known as descriptive assignment or equal times journey principle. The traffic flows that satisfy this principle are usually referred to as user equilibrium or user optimum flows, a term attributed to Dafermos in [44], since the routes chosen by the network users are those which are individually perceived to be the shortest under the prevailing conditions. The result from such individual decisions is a condition in which no user can reduce her travel time by choosing unilaterally another route, i.e., it is an equilibrium situation, similar to the Nash equilibrium introduced in Section 2, in a non-cooperative game where the players are associated with origin-destination pairs (see e.g. [123] for further references on the subject). Actually it will soon become apparent that, under specific assumptions, the game played is a team game and can therefore be stated as an optimization problem of type CPP.

By contrast, system optimum flows satisfy Wardrop’s second condition which states that “the average journey time is minimum”. These flow patterns are characterized by the fact that all routes used between an origin and a destination have equal marginal travel times, that is, the total travel time in the network is minimized, and this is considered as the system’s understanding of optimal network utilization. However, the total travel time is generally not minimized by the user optimal flows, and, moreover, observed flows in real life are closer to the user flow patterns than system optimum. The only situation in which the two flow patterns are equal is in the absence of congestion; this is an ideal case of course. In both system and user optimum flows, the fundamental units are considered to be vehicles or, in the case of public transport, the individual travelers.

Traffic equilibrium problems are frequently divided into two modeling cases: fixed demand and elastic demand. The additional case of stochastic demand will not be discussed here (see e.g. [135]). In the fixed demand case, an origin-destination demand matrix $R = \left[ r_k \right]$, with $r_k$ being the travel demand between the $k$th origin-destination pair, is assumed given. By contrast, in the elastic demand case, the demand $r_k$ is modeled as a function of the least travel cost between the end points of the $k$th origin-destination pair. Thus, the user has a number of travel choices available and she is economically motivated in her decision of making or not the trip.

In telecommunication networks, and especially in data and computer net-
works, where the routed units are packets, the routing has historically been based on the minimization of total average time \[21, 20, 59, 65, 72, 147, 148\], typically under the assumption of Poisson arrivals and exponential message lengths, and the independence assumption of Kleinrock \[93, 94\] for the M/M/1 queues of packets. However, equilibrium flow patterns have increasingly drawn the attention of the researchers in recent years. Indeed, the equilibrium principle of Wardrop can be adapted to model the situation where the routing decisions are made by the nodes instead of the users of the network in order to minimize the per packet delay. Actually, the routers at the nodes attempt to minimize the per packet delay in terms of hops, i.e., nodes, to the destination. In ad-hoc networks, however, where both the users and the base stations are mobile incurring constantly changing delays, it has been argued \[76\] that actual delays instead of hops should be considered. In such cases, the Wardrop equilibrium principle can be utilized to describe the resulting flow patterns. Wardrop equilibria have also been used in telecommunication networks where users can individually determine their route in order to route entire sessions instead of packages \[99\]. Wardrop’s equilibrium principle is also applicable to the case of distributed computer networks where the routed entities are entire jobs, and an individual job can be processed in any of the interconnected computers. In such a case, the routing decisions concern the minimization of the expected communication and processing delay in the system (see e.g. \[2\] for further references).

Let \( G = (A, N) \) be the underlying network with \( N \) the set of nodes and \( A \) the set of links, \( K \subseteq N \times N \) the set of origin-destination pairs, \( P_k \) the set of simple paths (routes) between the end nodes of the \( k \)th origin-destination pair, \( c_{pk} \) and \( h_{pk} \) the travel time and flow respectively on the \( p \)the route in \( P_k \).

According to Wardrop’s equilibrium principle, if \( \pi_k \) denotes the shortest route travel time between the end nodes of the \( k \)th origin-destination pair, then
\[
\begin{align*}
h_{pk} > 0 \Rightarrow c_{pk} &= \pi_k, \forall p \in P_k, \\
h_{pk} = 0 \Rightarrow c_{pk} &\geq \pi_k, \forall p \in P_k,
\end{align*}
\]
hold for all pairs in \( K \). Thus, including flow feasibility constraints, the user equilibrium conditions for fixed demand can be stated as follows:
\[
h_{pk}(c_{pk} - \pi_k) = 0, \forall p \in P_k,
\] (21)
\[ c_{pk} - \pi_k \geq 0, \quad \forall p \in \mathcal{P}_k, \quad (22) \]
\[ \sum_{p \in \mathcal{P}_k} h_{pk} = r_k, \quad (23) \]
\[ h_{pk} \geq 0, \quad \forall p \in \mathcal{P}_k \quad (24) \]
\[ \pi_k \geq 0, \quad (25) \]

for all origin-destination pairs \( k \).

For every link \( a \in \mathcal{A} \), let \( x_a \) denote the total link flow, and let \( s_a(x_a) \) be the link travel cost encountered by a user traveling on link \( a \) with a total flow \( x_a \). Define the link-route incidence matrix \( \Delta = [\delta_{kap}] \), where \( \delta_{kap} \) is 1 if the route \( \rho \) of the \( k \)th pair uses link \( a \), and 0 otherwise. We then have the following result [44]:

**Theorem 14** Assume that the network \( G = (\mathcal{N}, \mathcal{A}) \) is strongly connected with respect to the pairs in \( K \), that the demand matrix \( \mathbf{R} \) is nonnegative, and that the travel time function \( s_a(\cdot) \) is positive, strictly monotone increasing and continuously differentiable. Then, conditions (21)-(25) are the Karush-Kuhn-Tucker optimality conditions of the convex optimization problem

\[
(\text{FTAP}) \quad \min \sum_{a \in \mathcal{A}} \int_0^{x_a} s_a(t) dt,
\]
\[ \text{s.t.} \quad \sum_{p \in \mathcal{P}_k} h_{pk} = r_k, \quad \forall k, \quad (27) \]

\[ \sum_k \sum_{p \in \mathcal{P}_k} \delta_{kap} h_{pk} = x_a, \quad \forall a \in \mathcal{A}, \quad (28) \]
\[ h_{pk} \geq 0, \quad \forall p \in \mathcal{P}_k, \quad \forall k \quad (29) \]

By contrast, the system optimum seeking problem can be stated as follows:

\[
(\text{SFTAP}) \quad \min \sum_{a \in \mathcal{A}} s_a(x_a) x_a,
\]
\[ \text{s.t.} \quad (27) - (29) \]

Note that (28) are definitional rather actual constraints. Indeed, they can be eliminated by replacing \( x_a \) in the objective function (26) by the left-hand side in (28). Hence, both \text{FTAP} and \text{SFTAP} are defined over a Cartesian
product of \(|K|\) simplices, defined by (27) and (29), and are therefore special cases of CPP.

Typically, while in road networks the function \(s_a(x_a)\) has the form \(c_a + b_a \left(\frac{x_a}{u_a}\right)^\nu\), where \(c_a, b_a\) are constants specific to each link, \(c_a\) is the travel time on \(a\) at mean free speed, \(u_a\) is the practical capacity of the link, and \(\nu\) is some positive integer, in data communication networks, Kleinrock’s performance function \(s_a(x_a) = \frac{1}{u_a - x_a} + \mu t_a\) is used. Here \(u_a\) is the capacity (in bps) on link \(a\), \(t_a\) represents the processing delay, propagation delay or some other fixed delay in link \(a\), \(\frac{1}{\mu}\) is the average message length, assumed equal for all arcs, and \(x_a\) is the average flow (in bps). In both FTAP and SFTAP the objective functions are convex in route flows and strictly convex in link flows. This means that while the flow pattern in link flows is unique, that is, the total flow on each link has a unique optimal value, there may be a variety of optimal path flow patterns that all result in the unique link flow pattern. Thus, there are different optimal ways of of splitting flow from each origin to each destination among optimal routes.

To extend the user equilibrium model to the case of elastic demands, let \(r_k = g_k(\pi)\), where \(\pi = [\pi_1, \ldots, \pi_k, \ldots]\), that is, the travel demand \(r_k\) between the end nodes of the \(k\)th origin-destination pair is a function of the vector of the cheapest route costs. Then Wardrop’s user equilibrium principle for both route flows and demands are mathematically stated as follows:

\[
\begin{align*}
h_{pk} > 0 \quad & \Rightarrow \quad c_{pk} = \pi_k, \quad \forall p \in \mathcal{P}_k, \\
h_{pk} = 0 \quad & \Rightarrow \quad c_{pk} \geq \pi_k, \quad \forall p \in \mathcal{P}_k, \\
r_k > 0 \quad & \Rightarrow \quad r_k = g_k(\pi), \\
r_k = 0 \quad & \Rightarrow \quad g_k(\pi) \leq 0,
\end{align*}
\]

for all pairs \(k\). Introducing the flow feasibility requirements, the above equations lead, under the additional requirement of nonnegative \(g_k\) on the nonnegative orthant, to the following conditions:

\[
\begin{align*}
h_{pk}(c_{pk} - \pi_k) &= 0, \quad \forall p \in \mathcal{P}_k, \\
c_{pk} - \pi_k &\geq 0, \forall p \in \mathcal{P}_k, \\
\sum_{p \in \mathcal{P}_k} h_{pk} &= g_k(\pi), \\
h_{pk} &\geq 0, \forall p \in \mathcal{P}_k, \\
\pi_k &\geq 0,
\end{align*}
\]
for all origin-destination pairs $k$.

In their seminal work [8], Beckmann et al. recognized in (30)-(34) an optimization problem. Indeed, assume that $g_k$ has the additional property of being continuous and strictly increasing. Then it is invertible, in which case $r_k = g_k(\pi_k) \Leftrightarrow \pi_k = g^{-1}(r_k)$, whenever $r_k > 0$. Thus, we have the following result:

**Theorem 15** Assume that the network $G = (\mathcal{N}, \mathcal{A})$ is strongly connected with respect to the pairs in $\mathcal{K}$, and that the travel time function $s_a$ is positive, monotone increasing and continuous differentiable. Then, conditions (30)-(34) are the Karush-Kuhn-Tucker optimality conditions of the following convex optimization problem:

\[
\text{(ETAP)} \quad \min \sum_{a \in \mathcal{A}} \int_{0}^{x_a} s_a(t)dt - \sum_{k} \int_{0}^{r_k} g_k^{-1}(t)dt,
\]

s.t. \[
\sum_{p \in \mathcal{P}_k} h_{pk} = r_k, \quad \forall k, \tag{36}
\]

\[
\sum_{k} \sum_{p \in \mathcal{P}_k} \delta_{kap} h_{pk} = x_a, \quad \forall a \in \mathcal{A}, \tag{37}
\]

\[
h_{pk} \geq 0, \quad \forall p \in \mathcal{P}_k, \forall k, \tag{38}
\]

\[
r_k \geq 0, \quad \forall k. \tag{39}
\]

The objective function of ETAP is convex for any increasing $s_a(.)$ and any decreasing $g_k(.)$. Thus, ETAP consists of minimizing a convex function over linear constraints. Again, the definitional equations (37) can be removed by replacing $x_a$ in the objective function (35) by the left-hand side of (37). Hence, ETAP is a problem over the Cartesian product of the simplices (36), and therefore a special case of CPP.

The problems FTAP, SFTAP, and ETAP can be formulated in terms of only link flows instead of route flows. For this, let $o(k)$ and $d(k)$ denote the origin and the destination of each pair $k \in \mathcal{K}$, let $\mathcal{A}^+(i)$ be the set of links emanating from node $i \in \mathcal{N}$ and $\mathcal{A}^-(i)$ the set of links terminating at node $i \in \mathcal{N}$. Then, FTAP can be stated as:

\[
\text{(AFTAP)} \quad \min \sum_{a \in \mathcal{A}} \int_{0}^{x_a} s_a(t)dt, \tag{40}
\]

The objective function of AFTAP is convex for any increasing $s_a(.)$ and any decreasing $g_k(.)$. Thus, AFTAP consists of minimizing a convex function over linear constraints.
\[
\begin{align*}
\text{s.t.} \quad & \sum_{a \in \mathcal{A}^+(i)} x^k_a - \sum_{a \in \mathcal{A}^-(i)} x^k_a = r^k_i, \quad \forall i \in \mathcal{N}, \forall k \in \mathcal{K}, \quad (41) \\
x_a = \sum_{k \in \mathcal{K}} x^k_a, \quad \forall a \in \mathcal{A}, \quad (42) \\
x^k_a \geq 0, \quad \forall a \in \mathcal{A}, \forall k \in \mathcal{K}, \quad (43)
\end{align*}
\]

where \( x^k_a \) denotes the portion of flow from origin \( o(k) \) to destination \( o(k) \) that streams through link \( a \), and

\[
\begin{align*}
r^k_i &= \begin{cases} 
  r_k, & \text{if } i = o(k), \\
  -r_k, & \text{if } i = d(k), \\
  0, & \text{otherwise}.
\end{cases}
\end{align*}
\]

The corresponding system optimum flow assignment problem can then be stated as

\[
\text{(SAFTAP)} \quad \min \sum_{a \in \mathcal{A}} s_a(x_a)x_a,
\]
\[
\text{s.t. } (41) - (43).
\]

Similarly, \textbf{ETAP} can be restated as follows:

\[
\text{(AETAP)} \quad \min \sum_{a \in \mathcal{A}} \int_{0}^{x_a} s_a(t)dt - \sum_{k} \int_{0}^{r^k} g^{-1}_k(t)dt,
\]
\[
\text{s.t. } (41) - (43),
\]

where

\[
\begin{align*}
r^k_i &= \begin{cases} 
  r_k, & \text{if } i = o(k), \\
  -r_k, & \text{if } i = d(k), \\
  0, & \text{otherwise}.
\end{cases}
\end{align*}
\]

and \( r_k \geq 0 \). Constraints (41) are the so-called flow balance equation which state that flow is generated at each origin, absorbed at each destination, and is not destroyed in intermediate nodes. There is a separate set of such constraints for each origin-destination pair \( k \). Note also that all restated models are still special cases of \textbf{CPP}, where the Cartesian product is now with respect to the polyhedral sets defined by (41) and (43).

Clearly, in \textbf{ETAP} and \textbf{AETAP}, the demands of the network users are modeled so as to depend on the network state, which in turn depends on
the users’ choices of routes. Elastic demands have been studied extensively for road networks (see e.g. [128, 54, 66, 135]) but also in single commodity setting for trade networks, e.g. [74].

The differences between system and user optimization are the sources of several paradoxes [45] with most known that of Braess [32], which have been studied extensively for road networks [128] and have recently attracted the attention of the telecommunications community [99]. Braess’ paradox is illustrated in the next section.

3.2 Capacity Assignment and Network Design

The purpose of improving a given network by adding to its links more capacity or by adding to it entirely new links is to improve a certain traffic situation, that is, to decrease the delays in routing flows between remote locations. However, since routing according to the previous section can be done in two different and probably conflicting ways, i.e., from the users’ perspective or the system’s, and since the investment in the design or improvement of the network is undertaken by some central authority, one can suspect that conflicting situations may arise. In order to highlight this fact, consider the next example.

Example 1 Consider the network in Figure 1(a), where the link costs $s_{ij}$ are linear increasing functions of the flow $x_{ij}$ for all links $(i, j)$ of the network, and assume that there are 6 units of flow to be routed from node 1 to node 2. The total delay on each link $(i, j)$ for the user equilibrium model FTAP is then given by:

\[
\begin{align*}
    f_{13}(x_{13}) &= 5x_{32}^2 \\
    f_{32}(x_{32}) &= 50x_{32} + 0.5x_{32}^2 \\
    f_{14}(x_{14}) &= 50x_{14} + 0.5x_{14}^2 \\
    f_{42}(x_{42}) &= 5x_{42}^2
\end{align*}
\]

By inspection, the equilibrium flow pattern is given by routing 3 units through the path $1 \rightarrow 3 \rightarrow 2$ and 3 units through the path $1 \rightarrow 4 \rightarrow 2$ because of the symmetry. This equilibrium flow pattern generates a total (system) delay $\sum_{(i,j) \in A} s_{ij}(x_{ij}) = 498$, and the delay encountered by each unit of flow on each of the two paths is 83.

Suppose next that the network is being expanded by the addition of a new link $(3, 4)$ as shown in Figure 1(b). Then the equilibrium flow pattern

\[
\begin{align*}
    f_{34}(x_{34}) &= 5x_{34}^2 \\
    f_{43}(x_{43}) &= 50x_{43} + 0.5x_{43}^2 \\
    f_{23}(x_{23}) &= 5x_{23}^2
\end{align*}
\]
in the so-modified network is given by routing 2 units along the path 1 → 4 → 2, 2 units along the path 1 → 3 → 2, and 2 units along the new path 1 → 3 → 4 → 2. The new total (system) delay is then 552, that is, the intention of improving the network performance with the addition of a new link results in a worse situation than before since the total (system) delay has increased. Moreover, the delay of each flow unit along the two old paths has now increased to 92. ■

![Figure 1: Illustration of Braess’ paradox in network design](image)

We may now conclude that in the context of network design/improvement, we cannot be sure that the addition (or even deletion) of a link to (from) an existing network will not increase (decrease) the total delay at equilibrium unless, somehow, the users’ behavior has been taken into consideration in conjunction with the new network infrastructure. The phenomenon in which the equilibrium flow in an augmented network yields an increase in origin-destination travel times is called the Braess paradox. A direct consequence of its occurrence is that, if the equilibrium principle of Wardrop is adopted to represent the behavior of the network users, restriction or suppression of travel on some of the network links may reduce not only the total (system) delay, but also the delay of each individual flow unit. Possible occurrences of the paradox have been reported for road networks in Europe [97] and in USA [12].

The network design problem from the system’s perspective, i.e., ignoring user behavior, can be stated mathematically by slightly modifying the system flow assignment models of the previous section. Let $y_a$ be the decision variable of the design/improvement of link $a$. We will assume that $y_a$ mainly concerns assigning link $a$ some capacity level and that capacity comes in arbitrarily divisible small units, that is $y_a$ is a continuous and not a discrete
variable. Moreover, let $\mathcal{Y}_a$ be the set from which $y_a$ takes its values. Clearly a value $y_a = 0$ implies that the corresponding link is suppressed. We will further assume that the investment to a capacity level $y_a$ for link $a$ incurs a cost $g_a(y_a)$, where $g(\cdot)$ is some continuous function, and that the delays $s_a$ are functions of the the capacity level $y_a$ as well as of the total flow $x_a$ routed through link $a$. Then, a typical network problem can be stated as:

$$\text{(SNDP)} \quad \min \sum_{a \in \mathcal{A}} \{s_a(x_a, y_a)x_a + g_a(y_a)\}$$

s.t. $\quad (27) - (29)$ or, equivalently, $(41) - (43)$

$$y_a \in \mathcal{Y}_a, \forall a \in \mathcal{A}$$

It should be noted that in data communication networks, explicit capacity constraints on the flow are often present $[73, 21]$, that is,

$$x_a \leq u_a + y_a, \forall a \in \mathcal{A}$$

should be included in the above formulation.

If the functions $g_a(y_a)$ are convex nonlinear or linear, then this is a problem which is not more difficult to solve than the problems of the previous section. However, typically these are not always very realistic assumptions $[153]$, and, in particular, convexity of the functions would in general result in small capacity improvements in almost all links. The presence of two blocks of variables encourages then the application of a so-called primal or Benders decomposition $[68, 69, 127]$ to the problem in which, the $|\mathcal{A}|$ capacity assignment subproblems $[21, 153, 148]$ are solved and a so-called master problem, which is a flow assignment problem, is formed

$$f_a(x_a) = \min_{y_a} s_a(x_a, y_a)x_a + g_a(y_a)$$

s.t. $y_a \in \mathcal{Y}$

are solved and a so-called master problem, which is a flow assignment problem, is formed

$$\text{(BMP)} \quad \min \sum_{a \in \mathcal{A}} f_a(x_a)$$

s.t. $\quad (27) - (29)$ or, equivalently, $(41) - (43)$

If all $g_a(y_a)$ have some particular form, then $f_a(x_a)$ could be obtained in closed form. The case of linear $g_a(y_a)$ is discussed in $[73]$. However, this not
the case in general. Moreover, in the presence of capacity constraints, \( f_a(v_a) \) turn concave even if \( g_a(y_a) \) are linear. Even worse, if \( g_a(y_a) \) are not linear, \( f_a(x_a) \) may not even be differentiable [153].

Since in many situations the network manager or planner cannot impose the routing strategy of the network users, she must rather rely on some oracle to predict the user equilibrium flow patterns that will be the result of her decisions. Here lies the danger of the occurrence of Braess and other paradoxes. Thus, this case requires special modeling attention. The bilevel modeling of the Stackelberg game provides the way:

\[
\begin{align*}
\text{(SNDP)} & \quad \min_y \sum_{a \in A} \{ s_a(x^*_a, y_a) v_a + g_a(y_a) \} \\
\text{s.t.} \quad & y_a \in \mathcal{Y}_a, \forall a \in A \\
& \text{where } x^* = [x_a]_{a \in A} \text{ solves} \\
& \min_x \sum_{a \in A} \int_0^{x_a} s_a(t, y_a) dt \\
& \text{s.t. } (27) - (29) \text{ or, equivalently, } (41) - (43)
\end{align*}
\]

Clearly, the leader minimizes total system delay time and total investment subject to the restriction that the calculation of the total system delay is based on the equilibrium flow patterns and not on system routing flow patterns. SNDP can be restated as a nonlinear programming problem, if the second level problem is replaced by its optimality conditions corresponding to (21)-(25). However, the feasible region resulting from such a transformation is not a convex set and therefore the methods of Section 4 are, in general, unable to solve SNDP. See, for instance, [122, 123] and the references therein for further analysis of the problem.

### 3.3 Convex problems in supply chain optimization

We model here three problems that may arise in the context of supply chain management and which have the form of minimization of a convex differentiable function subject to linear constraints.
3.3.1 The transportation problem with customer competition for the offered service level

In this subsection, we assume that a producer, who owns several facilities produces/supplies a product to a set of customers. Customers may be either final users of the product or intermediate producers who use the product during their production process. We assume that customers themselves have to transport the product in order to satisfy their demand. Therefore, the customers incur transportation costs.

We assume further that the producer tries to provide its customers the best, at his opinion, level of service at minimum cost. The level of service provided is measured by the service delay encountered at any facility. The customers on the other hand, compete with each other for the provided service in order to satisfy their demand and make decisions based on the minimization of their individual total cost which includes not only the price of the product and the transportation cost but also the costs incurred due to delay observed in the service process. Thus, customer competition can be expressed in terms of a noncompetitive Nash game [129].

Let \( n \) be the number of customers who have demands on some product available through \( m \) warehouses. Let \( r_j \) be the demand of customer \( j \), \((j = 1, \ldots, n)\) and \( q_i \) the capacity of warehouse \( i \), \((i = 1, \ldots, n)\). We denote by \( x_{ij} \) the quantity purchased by customer \( j \) at facility \( i \). Typically, it is assumed that the cost encountered by customer \( j \) in purchasing a unit from warehouse \( i \) is a constant \( t_{ij} \), which is usually termed transport cost. The classical transportation problem is then formulated as follows:

\[
\text{(TP)} \quad \text{min} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} x_{ij} \quad (44)
\]

\[
\sum_{i=1}^{m} x_{ij} = r_j, \ \forall j \quad (45)
\]

\[
\sum_{j=1}^{n} x_{ij} \leq q_i, \ \forall i \quad (46)
\]

\[
x_{ij} \geq 0, \ \forall i, \ \forall j \quad (47)
\]

However, the assumption of the constant costs may not apply when customers are those traveling in order to receive the service they require since
their decision is influenced not only by the transportation costs to the facility but also by the time spent in it. Typical example such of cases are customers of commercial centers, of cash and carry centers, petrol stations etc. In all these cases the cost of transportation is transferred from the facilities to the customers who face the additional costs of the delay in their demand satisfaction. Thus, customers should be considered as rational decision makers who attempt to optimize their utility and whose decisions are affected by both the transportation cost and the services provided by the facility. Thus, a customer will choose to buy from a specific facility taking into account the price of the product, the transportation cost, and the expected waiting cost for service. The service time of a customer in a facility is affected by both his own decision and the choices of the others. In fact, the more customers choose a particular facility to be served the greater the waiting cost may turn for each one of them, as congestion phenomenon can appear.

We assume that customers have complete information of the situation at each facility and that based on this information each one decides from which one to satisfy his demand. Let the total amount serviced by warehouse $i$ be

$$x_i = \sum_{j=1}^{n} x_{ij}. \tag{48}$$

Then, $d_i(x_i)$ is the delay faced by customer $j$ at facility $i$ for each unit served. That is, it is a function of the total amount served by facility $i$. The function $d_i(x_i)$ may be considered as a performance function which measures the level of service offered by facility $i$. It is considered to be a differentiable and increasing function of $x_i$. The performance function is asymptotic to a specific level of product corresponding to the capacity of the facility. The level of service at the facility $i$ depends only on the load faced by this particular facility, i.e. it holds that:

$$\frac{\partial d_i(x_i)}{\partial x_k} = 0 \quad \forall i \neq k \tag{49}$$

$$\frac{\partial d_i(x_i)}{\partial x_i} > 0 \quad \forall i \tag{50}$$

For instance, in case of Poisson arrivals, if Kleinrock’s independence approximation holds, the perceived delay per serviced unit can be shown to be

$$d_i(x_i) = \frac{1}{q_i - x_i} \tag{51}$$
The perceived unit cost will therefore be in general of the form

\[ c_{ij}(x_i) = \alpha_i p_i + \beta_{ij} t_{ij} + \mu_i d_i(x_i) \]  

(52)

where \( p_i \) is the unit price of the product offered at facility \( i \), \( t_{ij} \) is the perceived unit transport cost as function of the distance between facility \( i \) and customer \( j \)'s residence, \( d_i(x_i) \), is the delay per serviced unit, while \( \alpha, \beta \) and \( \mu \) are transforming all contributing factors into the same disutility units.

We consider first the case where customers are assigned for their demand satisfaction to the facilities by a central coordinator, the producer. In such cases, the coordinator will try to assign the customers to the facilities in a manner that minimizes the average cost, or, equivalently, the total cost. Thus, the problem to be solved is as follows:

\[
\text{(SO - TP)} \quad \min \sum_{i=1}^{m} d_i(x_i)x_i + \sum_{i=1}^{m} p_i x_i + \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} x_{ij} \tag{53}
\]

s.t.

\[
\sum_{i=1}^{m} x_{ij} = r_j, \quad \forall j \tag{54}
\]

\[
x_i \leq q_i, \quad \forall i \tag{55}
\]

\[
x_i - \sum_{j=1}^{n} x_{ij} = 0, \quad \forall i \tag{56}
\]

\[
x_{ij} \geq 0, \quad \forall i, \quad \forall j \tag{57}
\]

The objective function of the problem (53) minimizes the total cost of delay plus cost of transportation and purchasing cost. Constraints (54) ensure that the total amount purchased by customer \( j \) meet his overall demand. Constraints (55) impose that the total amount available from each facility \( i \) does not exceed its capacity, while relations (56) ensure the maintenance of the flow in the network. It can be proved that the problem SO-TP described by (53)-(57) is a convex optimization problem with a unique solution [89].

In this system optimum problem, the central coordinator in order to optimize the benefits of the system, i.e. to minimize the average cost for all users, will assign customer \( j \) to facility \( i \) only if the total unit cost \( \tilde{c}_{ij}(x_i) \) faced at \( i \) is equal to the minimum marginal cost. That is, in order to take place a transaction between customer \( j \) and facility \( i \) the following must hold:

\[
x_{ij} > 0 \quad \Rightarrow \quad d_i(x_i) + x_i \frac{\partial d_i(x_i)}{\partial x_i} + p_i + t_{ij} = \tilde{c}_{ij} \quad \forall i, j \tag{58}
\]
\[ x_{ij} = 0 \Rightarrow d_i(x_i) + x_i \frac{\partial d_i(x_i)}{\partial x_i} + p_i + t_{ij} \geq \bar{c}_{ij} \quad \forall i, j \] (59)

Hence, problem (53)-(57) can be expressed as a nonlinear complementarity problem:

\[ \left[ d_i(x_i) + x_i \frac{\partial d_i(x_i)}{\partial x_i} + p_i + t_{ij} - \bar{c}_{ij} \right] x_{ij} = 0, \quad \forall i, \forall j \] (60)

\[ d_i(x_i) + x_i \frac{\partial d_i(x_i)}{\partial x_i} + p_i + t_{ij} - \bar{c}_{ij} \geq 0, \quad \forall i, \forall j \] (61)

\[ \sum_{j=1}^{n} x_{ij} = x_i \leq q_i, \quad \forall i \] (62)

\[ \sum_{i=1}^{m} x_{ij} = r_j, \quad \forall j \] (63)

\[ x_{ij} \geq 0, \quad \forall i, \forall j \] (64)

If we introduce Lagrangian multipliers \( \pi_i \geq 0 \) for restrictions (55) and \( w_j \leq 0 \) for restrictions (54), it can be proved [89] that conditions (60-64) are identical to the Karush Karush-Kuhn-Tucker optimality condition of the problem (53)-(57) for \( \bar{c}_{ij} = w_j - \pi_i \).

The multipliers \( w_j \) express the cost the customer \( j \) is willing to take on in order to increase his demand by one unit. While the multipliers \( \pi_i \) express the amount the facility \( i \) is willing to spend in order to increase its capacity by one unit. Therefore, the difference \( w_j - \pi_i \) is the profit or loss gained by facility \( i \) if it will increase its capacity by one unit. It is understood that facility \( i \) has an incentive to increase its capacity only if \( w_j - \pi_i \geq 0 \) that is, it has profit. The multipliers \( \pi_i \) have an additional interesting economic interpretation. They reflect the service time gained by the customers at a facility \( i \) if it increases its capacity and consequently its service rate. Therefore, the difference \( \bar{c}_{ij} = w_j - \pi_i \) can also express the minimum marginal cost faced by the customer \( j \) at facility \( i \).

The optimal quantities \( x_{ij} \) resulting from the problem SO-TP cannot in general represent an equilibrium state that minimizes the costs of each individual customer. In other words, customers can reduce their incurred cost if unilaterally change the facility from which they are served.

We consider next, the case where customers decide individually from which facility they will be served. In such a case, each customer will try
to assign himself for service to the facility he considers more profitable to himself and he will do so, antagonistically, if necessary, with the other customers. To model this case, let $\hat{c}_{ij}$ be the maximum unit cost at which customer $j$ is willing to buy. We then have for the customer and each facility:

$$x_{ij} > 0 \Rightarrow d_i(x_i) + p_i + t_{ij} = \hat{c}_{ij} \quad \forall i, j \quad (65)$$

$$x_{ij} = 0 \Rightarrow d_i(x_i) + p_i + t_{ij} \geq \hat{c}_{ij} \quad \forall i, j \quad (66)$$

That is, customer $j$ assigns himself for service to facility $i$, if the perceived unit cost $c_{ij}(x_i)$ for that facility is equal to the maximum unit cost $\hat{c}_{ij}$ at which customer $j$ is willing to buy. Moreover, all facilities selected to service customer have necessarily marginal costs equal to $\hat{c}_{ij}$. From equation (66), it follows that a facility $i$ not selected for service by customer $j$ has necessarily a marginal cost not lower than the maximum unit cost at which customer $j$ is willing to buy.

Introducing the feasibility conditions imposed by customer demands and facility capacities, the problem can then be stated as the following nonlinear complementarity problem:

$$[d_i(x_i) + p_i + t_{ij} - \hat{c}_{ij}] x_{ij} = 0, \quad \forall i, \forall j \quad (67)$$

$$d_i(x_i) + p_i + t_{ij} - \hat{c}_{ij} \geq 0, \quad \forall i, \forall j \quad (68)$$

$$x_i \leq q_i, \forall i \quad (69)$$

$$x_i - \sum_{j=1}^{n} x_{ij} = 0, \quad \forall i \quad (70)$$

$$\sum_{i=1}^{m} x_{ij} = r_j, \quad \forall j \quad (71)$$

$$x_{ij} \geq 0, \quad \forall i, \forall j \quad (72)$$

It can be proved [89] that conditions (67)-(72) are the Karush-Kuhn-Tucker conditions of the following convex optimization problem:

$$(\text{UO} - \text{TP}) \min \quad \sum_{i=1}^{m} \int_{0}^{x_i} d_i(t) \, dt + \sum_{i=1}^{m} p_i x_i + \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} x_{ij} \quad (73)$$
where \( \hat{c}_{ij} = w_j - \pi_i \), and \( w_j \) and \( \pi_i \) are the Lagrangian multipliers corresponding to constraints (75) and (74) respectively.

The difference \( w_j - \pi_i \) expresses again the profit or loss of the facility \( i \) if it increases its capacity by one unit or the minimum marginal cost faced by customer \( j \) in facility \( i \).

### 3.3.2 The Stochastic Transportation Problem

The stochastic transportation problem was first introduced by Cooper and LeBlanc [43]. It is an extension of the classical transportation problem [5], where the demand \( r_j \) of the costumer \( j \) is uncertain, and thus, it could be treated as a continuous random variable with probability density function \( \varphi_j(r_j) \). Moreover, it is assumed that variables \( r_j \) are independent, that is, the demand of the costumer \( \ell \) does not affect the demand of costumer \( j \), for \( \ell \neq j \).

Let \( t_{ij} \) be the unit transportation cost from supply point \( i \) to demand point \( j \) and \( x_j = \sum_{i=1}^{m} x_{ij} \) the amount of the product dispatched to demand point \( j \). Since the demand is not known in advance, when a particular quantity \( x_j \) of product is transferred to one of the destinations the following cases may be faced:

1. \( x_j < r_j \), i.e., the product is under-supplied. This shortage creates a penalty \( g_j(r_j - x_j) \) caused by the lost sales.
2. \( x_j > r_j \), i.e., the product is oversupplied. The result of this surplus is a penalty which is equal to \( \omega_j(x_j - r_j) \).
3. \( x_j = r_j \), which does not entail any additional cost.

Thus, due to randomness of \( r_j \) the expected cost of not covering the
demand of customer $j$ is [1], [158]

$$
g_j \int_{x_j}^{\infty} (r_j - x_j) \varphi_j(r_j) \, dr_j, \quad (78)$$

while

$$
\varpi_j \int_{0}^{x_j} (x_j - r_j) \varphi_j(r_j) \, dr_j \quad (79)
$$

is the cost of the oversupplied demand.

Assuming further that each supply center has a production capacity of $q_i$ units, the problem can be stated as follows:

$$
\text{(STP) \quad \min} \sum_{j=1}^{n} \left\{ g_j \int_{x_j}^{\infty} (r_j - x_j) \varphi_j(r_j) \, dr_j + \varpi_j \int_{0}^{x_j} (x_j - r_j) \varphi_j(r_j) \, dr_j \right\} 
+ \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij} x_{ij} \quad (80)
$$

$$
\text{s.t} \sum_{j=1}^{n} x_{ij} \leq q_i, \quad \forall i \quad (81)
$$

$$
\sum_{i=1}^{m} x_{ij} = x_j, \quad \forall j \quad (82)
$$

$$
x_{ij} \geq 0, \quad \forall i, \forall j, \quad (83)
$$

Although the problem is considered as a convex optimization problem in many research works (see for example, [43], [108], [85],[84], none of them proves the problem’s convexity. Galego [62] provides a proof of convexity for the case of the Economic Lot Scheduling Problem with Random Demand. A direct proof of the convexity of (80)-(83) is provided by [89].

### 3.3.3 Production-transportation problem in the presence of an intermediary

We consider here the production-transportation problem in the presence of an intermediary. Similar problems have been proposed by [41]. Suppose that there exist $|M|$ supply points and $|N|$ demand points for a product. The supply points are controlled by $|K|$ producers ($M_k$ refers to production facilities owned by the $k^{th}$ producer) while the demand points are owned
by a single buyer. The buyer’s demand is a vector \( \mathbf{r} = [r_j] \). The buyer is addressed to an intermediary in order to satisfy this demand. In turn, the intermediary is able to contact the \(|K|\) producers to obtain the product and meet the buyer’s demand. Each producer \( k \) faces a different production function and charges a price for the product based on a pricing function \( o_k(\mathbf{s}_k) \), where \( \mathbf{s}_k = [x_{i}]_{i \in M_k} \) is a vector in \( \mathbb{R}^{|M_k|} \) containing the quantities produced by facility \( i \) owned by producer \( k \).

It is assumed that function \( p_k(\cdot) \) is a convex (non-decreasing) differentiable function with and that the function \( o_k(\cdot) \) is a convex (non-decreasing) differentiable function with \( o(0) = 0 \). Therefore, for a given quantity of demand \( \mathbf{r} \), the problem of the intermediary is to choose the quantities to be ordered from each manufacturer in order to meet the customer’s demand at minimum cost.

If we assume further that the transportation cost \( t_{ij} x_{ij} \), where \( t_{ij} \) denotes the unit transportation cost and \( x_{ij} \) represents the quantity transported from production facility \( i \) to demand point \( j \) is charged to the customer, then the problem of the intermediary can be stated as follows:

\[
\text{[ITP]} \quad \min \quad \sum_{k=1}^{|K|} o_k(\mathbf{s}_k) + \sum_{i=1}^{|M|} \sum_{j=1}^{|N|} t_{ij} x_{ij} \quad (84)
\]

s.t \[
\sum_{j=1}^{|N|} x_{ij} = x_i, \quad \forall i \\
\sum_{i=1}^{|M|} x_{ij} = r_j, \quad \forall j \\
x_{ij} \geq 0, \quad \forall i, \forall j \quad (85, 86, 87)
\]

Next, we extend the problem to the case where at each demand point \( j \) there exist \(|L|\) buyers. The problem has been discussed in details by [89]. In this case the demand faced by the intermediary is a vector \( \mathbf{r} = [r_{j}^{\ell}] \), where \( r_{j}^{\ell} \) is the demand of buyer \( \ell \) at demand point \( j \). Let \( x_i = \sum_{\ell=1}^{|L|} x_{i}^{\ell} \). Functions \( o_k(\cdot) \) and \( p_k(\cdot) \) are considered to be convex (non-decreasing) and differentiable. Therefore, for a given quantity of demand \( \mathbf{r} \), the problem of the intermediary is to choose the quantities to be ordered from each manufacturer in order to meet the customers’ demand at minimum cost. The problem can be
mathematically modeled as:

\[
\text{(MITP)} \quad \min \sum_{k=1}^{K} o_k(s_k) + \sum_{i=1}^{M} \sum_{j=1}^{N} \sum_{\ell=1}^{L} t_{ij} \left( \sum_{\ell=1}^{L} x_{ij} \right) \tag{88}
\]

\[
\text{s.t.} \quad \sum_{j=1}^{N} \sum_{\ell=1}^{L} x_{ij} = x_i, \ \forall i \tag{89}
\]

\[
\sum_{i=1}^{M} x_{ij}^\ell = r_j^\ell, \ \forall j, \ \forall \ell \tag{90}
\]

\[
x_{ij}^\ell \geq 0, \ \forall i, \forall j \tag{91}
\]

If we assume that each producer is able to charge a different price for every buyer, that is if \( o_k(s_k) \) and \( p_k(s_k) \) are separable with respect to \( \ell \) then the problem can be separate into \( \ell \) problems one for each buyer. It is therefore reasonable to consider that these two functions are non-separable in \( \ell \). Constraints (90) correspond to a Cartesian product. Finally is should be mentioned that the inseparability of the problem is caused by the non-separability of the objective function (88).

### 3.4 Other Applications

We briefly survey a few more telecommunications applications that have been formulated either as nonlinear optimization problems and/or have been based on game theoretic concepts similar to those encountered in road networks.

In the models of the previous sections, the route costs are additive, that is, the cost of any path is calculated as the sum of the costs on the links that form it, there are cases, however, where such an assumption does not apply. For instance, in the so-called power criterion, which is frequently used in flow control problems, the ratio between some power of the overall throughput to overall delay is used instead of link delays. In the case of road networks, equilibrium flow models for non-additive costs are investigated in [60, 61, 14], while [3] discuss a simplified case of equilibrium routing in telecommunications networks.

The Braess paradox has been studied in the context of queuing networks [7, 34, 42], in capacity allocation to communication networks [98, 99], in distributed computing [87], and in flow routing [146, 145]. Stackelberg strategies in telecommunications are considered in [145, 162, 98].
In [91] the issues of charging, rate control and routing for telecommunication networks, such as ATM networks, with elastic traffic are studied in a game theoretic setting and nonlinear optimization is employed.

Energy conserving routing is formulated as an optimization problem in [39]. Similarly, the problem of extending network life time in power constrained networks can be formulated and solved as a nonlinear optimization problem. The transmission rate assignment problem as well as other problems are formulated and solved as a nonlinear optimization problem in [13].

4  Algorithms

4.1  The Frank-Wolfe Algorithm

Consider the problem of minimizing a differentiable function over a polyhedral set, i.e.,

\[(P1) \quad \min \ f(x) \quad \text{s.t.} \quad x \in \mathcal{X},\]

where \(\mathcal{X} = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}\) is a nonempty polyhedron.

The algorithm of Frank and Wolfe [57] also known as the conditional gradient algorithm [142], originally stated for a quadratic objective function \(f(x) = q^T x + x^T Qx\), is one of the most popular techniques for the solution of certain instances of the nonlinear program \(P1\). The popularity of this technique is due in part to its ability to exploit special constraint structures such as network structures (e.g. [23, 36, 46, 47, 59, 72, 107]), and in part to the fact that it decomposes non-separable problems over Cartesian product sets (cf. [103, 122, 126]). Also its implementation is in general quite simple. The algorithm was proposed as a routing technique in data networks under the name of flow deviation method by Fratta et al. [59], and independently for the traffic assignment problem in road networks by Leblanc et al. [107].

The algorithm is based on the linearization of the objective function. That is, given an iteration point \(x^k \in \mathcal{X}\), the algorithm approximates the objective with a first order Taylor expansion at \(x^k\), resulting in the linear programming subproblem

\[(FW-SUB^k) \quad \min \ \nabla f(x^k)^T x \quad \text{s.t.} \quad x \in \mathcal{X},\]
where the constant terms have been dropped from the objective function. The solution $\bar{x}^k$ of this subproblem is used in the construction of the search direction of descent $d^k = \bar{x}^k - x^k$. A line search on the interval $[0, 1]$ furnishes the next iterate $x^{k+1}$, that is, $x^{k+1} = x^k + \alpha_k d^k$, where $\alpha_k \in \arg \min_{\alpha \in [0,1]} f(x^k + \alpha d^k)$, and the process is repeated.

If $f$ in $P1$ is convex, then an interesting aspect of the application of the Frank-Wolfe algorithm to $P1$ is the generation of a lower bound on $f(x^*)$ at each iteration point $x_k$, namely $f(x_k) + \nabla f(x_k)^T(x_k - \bar{x}^k)$. This is a consequence of Theorem 8 and the minimization in $FW-SUB^k$. However, these lower bounds are not monotonically increasing. Hence, at iteration $k$, the current lower bound is defined as $\text{lbd}^k = \max\{\text{lbd}^{k-1}, f(x_k) + \nabla f(x_k)^T(\bar{x}^k - x^k)\}$, where $\text{lbd}^{k-1}$ is the incumbent lower bound, initially set to $-\infty$. In practice, the algorithm can be terminated once $f(x_k) - \text{lbd}^k \leq \epsilon_1$ or $\frac{f(x_k) - \text{lbd}^k}{f(x_k)} \leq \epsilon_2$ for suitably chosen $\epsilon_1 > 0$ and $\epsilon_2 > 0$.

We observe further that due to the linearity of the objective function in $FW-SUB^k$, the subproblem separates into $n$ problems, one for each factor in the Cartesian product when the algorithm is applied to the team game $CPP$:

$$\begin{align*}
\min & \quad \nabla_i f(x_k)^T x_i \\
\text{s.t.} & \quad x_i \in X_i \\
& \quad \forall i \in N,
\end{align*}$$

(92)

where $\nabla_i f(x)$ denotes the gradient of $f$ with respect to the $i$th block of $x = [x_1, \ldots, x_i, \ldots, x_n]$ and it is itself the $i$th block component of $f$’s gradient $\nabla f(x) = [\nabla_1 f(x), \ldots, \nabla_i f(x), \ldots, \nabla_n f(x)]$. Clearly, the application of the Frank-Wolfe algorithm to $CPP$ acts as a parallel decomposition scheme since the subproblems (92) are not interacting.

The application of the algorithm to the flow and traffic assignment problems $FTAP$, $AFTAP$, $STAP$, and $SAFTAP$ in Section 3 results in subproblems $FW-SUB^k$ or, equivalently, (92) which are so-called all-or-nothing assignment problems. That is, the shortest path $p_\kappa$ between each origin-destination pair $\kappa$ is calculated in terms of the linearized costs $\nabla_\kappa f(x_k)$ at the current flow situation $x_k$, and the entire flow volume $r_\kappa$ is assigned to this path, resulting into an extreme flow pattern $\bar{x}^k$ which solves $FW-SUB^k$ or, equivalently, (92). Thus, the character of the subproblems, and actually of the overall algorithmic approach, is independent of whether the problem is stated in terms of link or path flows. The application of the algorithm to problems with elastic demand, $ETAP$ and $AETAP$, requires some additional precautions and modifications which are discussed in detail by [65, 66].
Evan’s algorithm \cite{54}, although referred to as a partial linearization algorithm, is essentially more of a very specialized realization of the Frank-Wolfe algorithm than of the partial linearization discussed in Section 4.2.

For concave $f(\cdot)$ in P1, the Frank-Wolfe algorithm without line search, i.e., $\alpha_k = 0$, $\forall k$, acts as a heuristic approach able to produce local solutions to P1, since by Theorem 10, the concave function $f(\cdot)$ attains a global minimum among the extreme points of the polytope $\mathcal{X}$, and the Frank-Wolfe subproblem produces an extreme point optimal to the linearization of $f(\cdot)$ at $x^k$. This property of the method has been utilized in several applications, often within the scheme of a multi-start local search or some other metaheuristic, notably in misclassification minimization and data mining \cite{9, 10, 11, 25} and in flow routing with concave costs \cite{161, 163} as the result of the primal decomposition approach to the network design and capacity assignment problems of Section 3.

Several convergence results exist for the algorithm in the form given above as well as for several of its specializations (e.g. \cite{35, 116, 142, 46, 47, 78}). For completeness of the exposition, we will demonstrate the global convergence of the approach to a global minimum point of P1 under the following assumptions:

\begin{enumerate}
\item[(A1)] \textit{$f(x)$ is continuously differentiable on $\mathcal{X}$,}
\item[(A2)] \textit{$f(x)$ is pseudoconvex on $\mathcal{X}$, and}
\item[(A3)] \textit{$\mathcal{X}$ is closed and bounded (i.e. a polytope).}
\end{enumerate}

Assumption A2 is essential for the verification of the global optimality of any accumulation point of the sequence \{$x^k$\}. Assumption A3 ensures that P1 and FW-SUB\(^k\) have finite optima. It can be replaced by the coercivity assumption \(\lim_{\|x\| \to \infty} f(x) = \infty\) on \(\mathcal{X}\), and the assumption that \(\nabla f(x^k)^T x\) is bounded from below on \(\mathcal{X}\) for all $x^k \in \mathcal{X}$.

\textbf{Theorem 16} \textit{Under the posed assumptions, the Frank-Wolfe algorithm either terminates finitely with an optimal solution of P1 or it generates an infinite sequence \{$x^k$\} of feasible points in $\mathcal{X}$ such that any of its accumulation points is an optimal solution of P1.}

\textbf{Proof:} The algorithm would terminate at iteration $k$ if the extreme point solution $\bar{x}^k$ of FW-SUB\(^k\) is such that $(\bar{x}^k - x^k)^T \nabla f(x^k) = 0$, implying
\[ \nabla f(x^k)^T \bar{x}^k = \nabla f(x^k)^T x^k. \] Indeed, the optimality of \( \bar{x}^k \) in \textbf{FW-SUB}^k implies 
\((\bar{x}^k - x)^T \nabla f(x^k) \leq 0, \forall x \in \mathcal{X}\), and therefore 
\((x - x^k)^T \nabla f(x^k) \geq 0, \forall x \in \mathcal{X}\). Consequently, 
\( x^k \) is optimum in \( P1 \). We therefore assume subsequently that 
\((\bar{x}^k - x^k)^T \nabla f(x^k) < 0 \). Letting \( \phi(\alpha) = f(x^k + \alpha(\bar{x}^k - x^k)) \), we observe that 
\( \phi'(0) = (\bar{x}^k - x^k)^T \nabla f(x^k) \), and therefore \( \phi'(0) < 0 \). Consequently, 
\( \phi(\alpha) < \phi(0) \) for some \( \alpha \in [0, 1] \) implying \( \phi(0) > \min_{\alpha \in [0,1]} \phi(\alpha) \), where the minimum exists by the continuity assumption. Thus, \( \phi(\alpha_k) < \phi(0) \) implying 
\( f(x^{k+1}) < f(x^k) \). Hence, \( \{x^k\} \) is a sequence of feasible points such that 
\( \{f(x^k)\} \) is a strictly decreasing sequence which is bounded from below 
due to the continuity of \( f(x) \) and the compactness of \( \mathcal{X} \). Consequently, 
\( \{f(x^k)\} \) has a finite limit \( f^* \) and the sequence \( \{x^k\} \) has an accumulation 
point \( x^* \in \mathcal{X} \) such that \( f(x^*) = f^* \). Assume that \( \{x^k\} \) is a subsequence 
such that \( \lim_{k \to \infty} x^k = x^* \). Then it must have a subsequence \( \{x^{\ell} \} \) such that the corresponding \( x^{\ell} \) is the same extreme point \( \bar{x} \) of \( \mathcal{X} \) for all \( k \). Such a subsequence exists since the number of extreme points is finite. Then, 
\( (\bar{x} - x)^T \nabla f(x^*) \leq 0, \forall x \in \mathcal{X}, \forall k \), which for \( k \to \infty \) implies 
\[ (\bar{x} - x)^T \nabla f(x^*) \leq 0, \forall x \in \mathcal{X}. \] (93)

Since \( \{f(x^k)\} \) is decreasing, for \( \ell > k \) we have \( f(x^\ell) \leq f(x^{k+1}) < f(x^k) \) which implies 
\( f(x^k + \alpha(\bar{x} - x^k)) \geq f(x^k) \) for some \( \alpha \in [0, 1] \). Taking the limit as \( k \) and \( \ell \) approach \( \infty \) we have \( x^k \to x^* \) and \( x^\ell \to x^* \), and consequently 
\[ \frac{f(x^\ell + \alpha(\bar{x} - x^\ell)) - f(x^\ell)}{\alpha} \geq 0 \] for \( \alpha \in [0, 1] \). Letting \( \alpha \to 0^+ \) we obtain (c.f. Theorem 6)
\[ (\bar{x} - x^*)^T \nabla f(x^*) \geq 0. \] (94)

Inequalities (93) and (94) imply 
\[ (x - x^*)^T \nabla f(x^*) \geq 0, \forall x \in \mathcal{X} \] and consequently \( x^* \) is optimum in \( P1 \). ■

Although the most attractive features of the Frank-Wolfe algorithm are inherent in the subproblems \textbf{FW-SUB}^k as they are linear and decomposable, these same subproblems are the sources of the disadvantages associated with the approach. Due to their linearity, they always produce extreme point solutions, and the directions obtained may therefore depend more on the properties of the feasible region \( \mathcal{X} \) than the properties of the objective function. It is also apparent from the convergence proof given above that the finiteness of the number of extreme points of \( \mathcal{X} \) imply repeated re-generation of the same extreme points within an infinite sequence and consequently a zig-zagging approach towards the optimal solution \( x^* \) should be expected.
and jamming as $\mathbf{x}^*$ is being approached. An extreme such case is given by the example below.

**Example 2** Consider the following instance of CPP:

$$
\begin{align*}
\text{min} & \quad (x_1 - 2x_4)^2 + (3x_2 - x_3)^2 + (x_1 - 2)^2 + (x_3 - 2)^2 + (x_2 - 5)^2 + (x_4 - 3)^2 \\
\text{s.t.} & \quad 3x_1 + 2x_2 \leq 6 \\
& \quad 5x_3 + 2x_4 \leq 10 \\
& \quad x_j \geq 0, \ j = 1, 2, 3, 4
\end{align*}
$$

Its optimal solution is $(1.407, 0.890, 1.583, 1.042)$ and the corresponding optimal objective function value is 22.89.

If the algorithm is initialized with the origin, it must approach the optimal solution by following directions which are based on the alternate generation of the two extreme points $(\bar{x}_1, \bar{x}_2) = (2, 0)$ and $(\bar{x}_3, \bar{x}_4) = (0, 3)$ of the first simplex and the alternate generation of the two extreme points $(\bar{x}_3, \bar{x}_4) = (2, 0)$ and $(\bar{x}_3, \bar{x}_4) = (0, 5)$ of the second simplex, zig-zagging thus towards the optimal solution. This phenomenon is illustrated in Figure 2. The jamming starts as the steplengths $\alpha_k$ become smaller for each new iteration. Table 1 shows a few iterations of the algorithm. Note the rapid decrease of the objective function value in early iterations and the small decimal changes in later iterations.

An asymptotic analysis of the Frank-Wolfe algorithm for maximum concurrent flows is presented in [23]. It can be shown that the convergence rate of the Frank-Wolfe algorithm is indeed sublinear [35, 52]. Thus, several improvements to the original algorithm, based either on the concept of feasible direction of descent or on the concept of column generation, have been proposed in recent years and these are examined in the following subsections.

### 4.2 Feasible Direction Improvements

The basic idea of improving the Frank-Wolfe algorithm while retaining its feasible direction nature is to not let the generated directions be based so heavily on the extreme points of the feasible region. This can be done basically in two different ways; either by avoiding the complete linearization of the objective function or by enriching the Frank-Wolfe subproblems with some nonlinear information. In both cases the original problem $\mathbf{P}$ is replaced
Figure 2: Movements of the Frank-Wolfe algorithm in the subproblem spaces iteratively by a sequence of easier (sub-) problems obtained by replacing the original objective function by a new one which may depend on the current iteration point.

It was proposed in [103] that the original function \( f(\cdot) \) should only be partially linearized, that is, if \( f(x) = \sum_{i=1}^{n} f_i(x_i) + e(x) \), where \( f_i(\cdot) \) are strictly convex functions and \( e(\cdot) \) is not additively separable, then only \( e(\cdot) \) needs to be linearized. Moreover, if \( f(\cdot) \) does not have the necessary form, such a form can be enforced with the introduction of a second function \( \varphi(\cdot) \), which may be assumed strictly convex and additively separable. Then, the original objective function \( f(\cdot) \) is replaced by the equivalent \( \varphi(\cdot) + [f(\cdot) - \varphi(\cdot)] \) and the “error” \( e(\cdot) = f(\cdot) - \varphi(\cdot) \) is linearized. In the case of problems whose feasible region is the Cartesian product of polyhedral sets, i.e., of the form of the CPP, the partial linearization approach retains the parallel decomposition property of the original Frank-Wolfe problem, replacing (92)
Table 1: Frank-Wolfe iterations

<table>
<thead>
<tr>
<th>$k$</th>
<th>$x^k$</th>
<th>$f(x^k)$</th>
<th>$x^*_1$</th>
<th>$x^*_2$</th>
<th>$\text{lb}d^k$</th>
<th>$d^k$</th>
<th>$\alpha_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0.000,0.000, 0.000)</td>
<td>42.000</td>
<td>(0.000,3.000)</td>
<td>(0.000,5.000)</td>
<td>-18.000</td>
<td>(0.000,3.000, 0.000)</td>
<td>0.139</td>
</tr>
<tr>
<td>2</td>
<td>(0.000,0.419, 0.000)</td>
<td>37.814</td>
<td>(2.000,0.000)</td>
<td>(2.000,0.000)</td>
<td>11.209</td>
<td>(2.000,-4.186, -2.000)</td>
<td>0.432</td>
</tr>
<tr>
<td>3</td>
<td>(0.763,0.562, 0.763)</td>
<td>29.102</td>
<td>(0.000,3.000)</td>
<td>(2.000,0.000)</td>
<td>22.290</td>
<td>(-1.284,2.174, 0.0179)</td>
<td>0.0179</td>
</tr>
<tr>
<td>33</td>
<td>(1.284,0.826, 1.284)</td>
<td>23.861</td>
<td>(0.000,3.000)</td>
<td>(2.000,0.000)</td>
<td>22.290</td>
<td>(-1.284,2.174, 0.0179)</td>
<td>0.0179</td>
</tr>
<tr>
<td>34</td>
<td>(1.261,0.865, 1.438)</td>
<td>23.847</td>
<td>(1.261,0.865)</td>
<td>(2.000,0.000)</td>
<td>22.290</td>
<td>(-1.284,2.174, 0.0179)</td>
<td>0.0179</td>
</tr>
</tbody>
</table>

by the strictly convex subproblems

$$
\min \left\{ f_i(x_j) + \nabla_i e(x^k)^T x_i : x_i \in \mathcal{X}_i \right\} \quad \forall i \in \mathcal{N}.
$$

Letting $\bar{x}^k = [\bar{x}^k_1, \ldots, \bar{x}^k_n]$ denote the point obtained by solving these subproblems, a feasible direction of descent $d^k = \bar{x}^k - x^k$ is formed and the next iterate $x^{k+1}$ is furnished by a line search on the interval $[0, \bar{\alpha}_k]$, where $\bar{\alpha}_k = \max\{\alpha \geq 0 : x^k + \alpha d^k \in \mathcal{X}\}$, that is, $x^{k+1} = x^k + \alpha_k d^k$, where $\alpha_k \in \arg\min_{\alpha \in [0,\bar{\alpha}_k]} f(x^k + \alpha d^k)$. The algorithm and its behavior is illustrated by the next example.

**Example 3** Partial linearization is applied to the problem instance of Example 2 by linearizing only the non-separable part of the objective function, i.e., $-4x_1x_4 - 6x_2x_3$, we then obtain the two subproblems:

=$((\text{SUB}^1)$) $\min \quad 2(x_1)^2 + 10(x_2)^2 - (4 + 4\bar{x}_4)x_1 - (10 + 6\bar{x}_3)x_2$

$s.t.$ $3x_1 + 2x_2 \leq 6$

$x_1, x_2 \geq 0$

and

=$((\text{SUB}^2)$) $\min \quad 2(x_3)^2 + 5(x_4)^2 - (4 + 6\bar{x}_2)x_3 - (6 + 4\bar{x}_1)x_4$

$s.t.$ $5x_3 + 2x_4 \leq 10$

$x_3, x_4 \geq 0$
Table 2 lists three iterations of the partial linearization algorithm, while Figure 3 illustrates the movements of the algorithm in the subproblems’ simplices. Comparing to the Frank-Wolfe algorithm in Example 2, we observe that the point obtained by the latter in the 34th iteration is further away from the optimal point than the first point produced by the partial linearization. Also from the figures it is clear that the partial linearization algorithm is less dependent on the shape of the feasible region since it retains more information about the original objective function in its subproblems.

One important property of the partial linearization algorithm, not shared by the Frank-Wolfe algorithm, is that the subproblems (95) are able to generate the optimal solution to the original problem.

In [122] the concept of regularization of the Frank-Wolfe subproblems was introduced. To large extend, this regularized Frank-Wolfe algorithm is based on the observation that although the linearization of $f(\cdot)$ at $x^k$ performed in (92) is mathematically good in some small neighborhood around $x^k$, subproblems (92) lack anything that would enforce such a restriction. Thus, [122] introduces a regularization function $\phi : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ satisfying the following properties:

(A4) $\phi$ is continuously differentiable on $\mathcal{X} \times \mathcal{X}$,

(A5) $\phi$ is nonnegative and convex on $\mathcal{X} \times \mathcal{X}$,

(A6) $\phi(x, y)$ is strictly convex for every fixed $y \in \mathcal{X}$,
Figure 3: Movements of the partial linearization algorithm in the subproblem spaces

(A7) $\phi(x, y)$ is strictly convex for every fixed $x \in \mathcal{X}$, and

(A8) $\nabla \phi(x, y) = [\nabla_x \phi(x, y), \nabla_y \phi(x, y)] = [0, 0]$ if and only if $x = y$.

Because of the last property, such functions can be thought of as “distance” functions which, however, need not satisfy symmetry and/or triangle inequality. Examples of functions that satisfy the stated requirements include the proximal point function $\phi(x, y) = \frac{1}{2}\|x - y\|^2$, the projection function $\phi(x, y) = \frac{1}{2}(x - y)^T D(x - y)$, where $D$ is a positive diagonal matrix, and the entropy function $\phi(x, y) = \sum_{i=1}^n \{(x_i + \epsilon) \ln \left(\frac{x_i + \epsilon}{y_i + \epsilon}\right) - (x_i - y_i)\}$, where $\epsilon > 0$ is a small constant.

In this regularized Frank-Wolfe approach, the subproblems (92) in the
original algorithm are replaced by

\[
\min \sum_i f(x^k)^T x_i + t_k \phi_i(x_i, x^k)
\]

\[\text{s.t. } x_i \in X_i \forall i \in \mathcal{N},\]

(96)

where \( t_k > 0 \) is some positive constant and \( \phi(x, x^k) = \sum_{i=1}^n \phi_i(x_i, x^k) \). The next iterate \( x^{k+1} \) is computed in the same manner as in the partial linearization algorithm. The algorithm shares with the partial linearization approach the property that the subproblems are able to generate the optimal solution of the original problem. Moreover, it can be shown that both algorithms converge globally to an optimal solution of \( P_1 \) or \( CPP \) under the conditions stated for the Frank-Wolfe algorithm. The proofs of convergence resemble that given above for the Frank-Wolfe algorithm ([122] and [103]). We present here these results for the regularized version of the algorithm.

In this approach, \( FW-SUB^k \) is replaced by the regularized subproblem

(\( RFW-SUB^k \)) \[
\min \sum_i f(x^k)^T x + t_k \phi(x, x^k)
\]

\[\text{s.t. } x \in X,\]

which, according to our assumptions, is a well-defined convex programming problem; its solution set is nonempty and singleton. The following two lemmas outline two important properties of the regularized subproblem.

**Lemma 1** If \( x^k \) solves \( RFW-SUB^k \), then \( x^k \) is also a solution to \( P_1 \). The converse is also true.

**Proof:** If \( x^k \) is optimal in \( RFW-SUB^k \), the variational inequality

\[
[\nabla f(x^k) + t_k \nabla x \phi(x^k, x^k)]^T (x - x^k) \geq 0, \forall x \in X
\]

(97)

is satisfied [92, Chapter I, Proposition 5.1]. By assumption \( A8 \), \( \nabla x \phi(x^k, x^k) = 0 \). Thus,

\[
\nabla f(x^k)^T (x - x^k) \geq 0, \forall x \in X.
\]

(98)

Hence, by assumption \( A2 \), \( x^k \) is optimal in the original problem \( P_1 \) [116]. Obviously, the reverse also holds true. \( \blacksquare \)

**Lemma 2** Let \( \hat{x}^k \) be the unique optimal solution in \( RFW-SUB^k \) and assume that \( \hat{x}^k \neq x^k \). Then \( d^k = \hat{x}^k - x^k \) is a feasible direction of descent.
Proof: The direction is feasible since \( \bar{x}^k \) and \( x^k \) are both in \( \mathcal{X} \), and the latter is a convex set.

By the optimality of \( \bar{x}^k \) in \( \text{RFW-SUB}^k \), the variational inequality

\[
[\nabla f(x^k) + t_k \nabla x \phi(\bar{x}^k, x^k)]^T(x - \bar{x}^k) \geq 0, \quad \forall x \in \mathcal{X}
\]

(99) holds. Thus, \( [\nabla f(x^k) + t_k \nabla x \phi(\bar{x}^k, x^k)]^T(x^k - \bar{x}^k) \geq 0 \), since \( x^k \in \mathcal{X} \).

This yields the inequality \( \nabla f(x^k)^T(x^k - \bar{x}^k) \geq -t_k \nabla x \phi(\bar{x}^k, x^k)]^T(x^k - \bar{x}^k) \) or

\[
\nabla f(x^k)^T(x^k - \bar{x}^k) \geq t_k \nabla x \phi(\bar{x}^k, x^k)]^T(x^k - \bar{x}^k). \quad (100)
\]

Assumption \textbf{A6} implies the strict monotonicity of \( \nabla x \phi \) (see e.g. [6], Theorem 3.3.4). Hence

\[
[\nabla x \phi(\bar{x}^k, x^k) - \nabla x \phi(x^k, x^k)]^T(\bar{x}^k - x^k) > 0,
\]

(101) since \( \bar{x}^k \neq x^k \). Due to assumption \textbf{A8}, this inequality reduces to

\[
\nabla x \phi(\bar{x}^k, x^k)^T(\bar{x}^k - x^k) > 0. \quad (102)
\]

Therefore, (100), by (102), yields \( \nabla f(x^k)^T(x^k - \bar{x}^k) > 0 \) and consequently \( \nabla f(x^k)^T(\bar{x}^k - x^k) < 0 \). This is the desired result. \( \blacksquare \)

We prove next the convergence of the feasible direction algorithm which utilizes the descent direction produced by the regularized subproblem.

\textbf{Theorem 17} The regularized Frank-Wolfe algorithm either terminates in a finite number of iterations or it generates an infinite sequence \( \{x^k\} \) such that any accumulation point is an optimal solution to \( \textbf{P1} \).

Proof: Due to Lemma 1, the algorithm terminates in a finite number of iterations whenever \( x^k = \bar{x}^k \), for some \( k \). We therefore assume that \( x^k \neq \bar{x}^k \) for all \( k \). It follows then, by Lemma 2, that the sequence \( \{d^k\} = \{\bar{x}^k - x^k\} \) consists of feasible directions of descent. Therefore, for \( \bar{\alpha}_k = \max\{\alpha|x^k + \alpha d^k| \in \mathcal{X}\} \) and \( \alpha_k \in \arg\min\{f(x^k + \alpha d^k)|\alpha \in [0, \bar{\alpha}_k]\} \), we have

\[
f(x^{k+1}) = f(x^k + \alpha_k d^k) < f(x), \quad (103)
\]

and \( \{f(x^k)\} \) is a monotonically decreasing sequence which is bounded from below due to assumptions \textbf{A1} and \textbf{A3}. According to the Weirstrass theorem, the sequence \( \{x^k\} \) must have at least one point of accumulation, say \( x^* \in \mathcal{X} \).
which is the limit of a convergent subsequence \(\{x^k\}_{k \in \mathcal{L}}\), \(\mathcal{L}\) being an infinite subset of \(\mathbb{N}\). We therefore have

\[
\lim_{k \to \infty, k \in \mathcal{L}} f(x^k) = f(x^*). \tag{104}
\]

From (104) it follows that

\[
\lim_{k \to \infty, k \in \mathcal{L}} [f(x^{k+1}) - f(x^k)] = 0,
\]

which due to (103) implies that

\[
\lim_{k \to \infty, k \in \mathcal{L}} [f(x^k + \beta_k(x^k - x^k)) - f(x^k)] \geq 0,
\]

for every sequence \(\{\beta_k\}_{k \in \mathcal{L}}\) such that \(\beta_k \in (0, \bar{\alpha}_k), \forall k\), and \(\lim_{k \to \infty, k \in \mathcal{L}} \beta_k = 0\). We thus have

\[
\lim_{k \to \infty, k \in \mathcal{L}} \frac{f(x^k + \beta_k(x^k - x^k)) - f(x^k)}{\beta_k} \geq 0. \tag{106}
\]

Since \(\{x^k\}_{k \in \mathcal{L}}\) has at least one accumulation point, say \(x \in \mathcal{X}\), which is the limit of a subsequence \(\{x^k\}_{k \in \mathcal{L}'}, \mathcal{L}' \subset \mathcal{L}\), the continuous differentiability of \(f\) implies through (106) that

\[
\nabla f(x^*)^T (x - x^*) \geq 0. \tag{107}
\]

By the optimality of \(\bar{x}^k\) in \textbf{RFW-SUB}^k we have

\[
[\nabla f(x^k) + t \nabla_x \phi(\bar{x}^k, x^k)]^T (x - \bar{x}) \geq 0, \forall x \in \mathcal{X}. \tag{108}
\]

Taking the limit over \(k \in \mathcal{L}'\) in (108) we get

\[
[\nabla f(x^*) + t \nabla_x \phi(\bar{x}, x^*)]^T (x - \bar{x}) \geq 0, \forall x \in \mathcal{X}. \tag{109}
\]

This yields

\[
[\nabla f(x^*) + t \nabla_x \phi(\bar{x}, x^*)]^T (x^* - \bar{x}) \geq 0. \tag{110}
\]

The inequalities (107) and (110) then imply the inequality

\[
\nabla_x \phi(\bar{x}, x^*)^T (x^* - \bar{x}) \geq 0. \tag{111}
\]

By assumption, \(\nabla_x \phi(\bar{x}, x^*) = 0\). Hence (111) implies

\[
[\nabla_x \phi(\bar{x}, x^*) - \nabla_x \phi(\bar{x}, x^*)]^T (x^* - \bar{x}) \geq 0. \tag{112}
\]

45
On the other hand, by the strict monotonicity of $\nabla_x \phi$, which is implied by (A6), we have
\[ [\nabla_x \phi(\bar{x}, x^*) - \nabla_x \phi(x^*, x^*)]^T (\bar{x} - x^*) > 0 \quad (113) \]
if $x^*/ = \bar{x}$. The relations (112) and (113) then imply that
\[ [\nabla_x \phi(\bar{x}, x^*)^T - \nabla_x \phi(x^*, x^*)]^T (\bar{x} - x^*) = 0. \quad (114) \]
The strict monotonicity of $\nabla_x \phi$ and (114) imply then that $\bar{x} = x^*$. Substituting $\bar{x} = x^*$ in (109) gives $\nabla f(x^*)^T(x - x^*) \geq 0, \forall x \in X$, as desired.

It is shown in [122] that the regularized Frank-Wolfe algorithm unifies under the same umbrella several known nonlinear programming algorithms, such as the Goldstein-Levitin-Polyak projection algorithm [110, 15, 17], the Newton method [16, 17, 96], and it also introduces several new methods, such as the partially linearized proximal point algorithm. Both partial linearization and regularized Frank-Wolfe are suitable for distributed computations [136], particularly as they converge, under suitable assumptions [90], for fixed step lengths $\alpha_k$. The subproblems in both algorithms can be solved in several different ways; approximate solutions based on the original Frank-Wolfe algorithm are discussed in [122], a Lagrangian relaxation approach along the lines of (9)-(10), which is in itself a fully distributed computation scheme [20], has been proposed in [105] and computational experience from its application to flow assignment problems are reported there, finally approaches similar to [96] or the more recent [28, 29, 30] can be adapted to solve the subproblems.

In the sequential implementation of the Frank-Wolfe algorithm, the subproblems (92) must be solved sequentially under some prespecified order, typically by increasing $i$. It is thus obvious that under such a realization of the Frank-Wolfe algorithm, although new information is generated during the loop of subproblem solving, the algorithm fails to take advantage of it during the duration of the loop. Thus, it is suggested in [126], under the name of cyclic linearization, that new information generated during the Frank-Wolfe subproblem solving loop (92) should be utilized during the duration of the loop. Hence, the loop (92) in the original algorithm is replaced by the following scheme:

\[
\begin{align*}
\bar{x}_i^k & \in \arg\min_{x_i \in X_i} \nabla_i f(x_i^{k+1}, x_i^k, x_{i+1}^k)^T x_i \\
\alpha_i^k & \in \arg\min_{\alpha \in [0,1]} f(x_i^{k+1}, x_i^k + \alpha(\bar{x}_i^k - x_i^k), x_{i+1}^k) \\
x_i^{k+1} & = x_i^k + \alpha_i^k(\bar{x}_i^k - x_i^k)
\end{align*}
\quad \forall i \in \mathcal{N} \quad (115)
\]

46
Finally several other improvements have been proposed by several authors (e.g. [108, 135, 156]). The PARTAN or parallel tangents approach has been suggested in particular in order to provide an acceleration scheme for the Frank-Wolfe algorithm that overcomes to certain extent the zig-zagging behavior. Thus, given two consecutive Frank-Wolfe iterates, \( x^k \) and \( x^{k+1} \), a line search is performed along the descent direction \( \bar{d}^{k+1} = x^{k+1} - x^k \) and the new point \( \bar{x}^{k+1} \) obtained as \( f(\bar{x}^{k+1}) = f(x^k + \alpha \bar{d}^{k+1}) = \min_{\alpha \in [0, \bar{\alpha}_k]} f(x^k + \alpha \bar{d}^{k+1}) \) replaces \( x^{k+1} \) in the next Frank-Wolfe iteration. It can be shown [112] that under certain conditions PARTAN has the behavior of the conjugate gradients method.

4.3 Simplicial Decomposition and Column Generation

Consider problem \( \textbf{P1} \). Since \( \mathcal{X} \) is a polyhedral set, by Caratheodory’s Theorem 4, any point in \( \mathcal{X} \) can be expressed as a convex combination of its extreme points plus a linear combination of its extreme directions. For simplicity of the exposition, let us assume that \( \mathcal{X} \) is a polytope and only use the first part of Theorem 4, otherwise, \( \mathcal{X} \) may be unbounded, in which case its extreme directions should be taken under consideration according to the second part of Theorem 4. Then \( \textbf{P1} \) can be restated in the equivalent form:

\[
\text{(MP1)} \quad \min_y f \left( \sum_{k=1}^{n} y_k \bar{x}^k \right),
\]

\[
\text{s.t.} \quad \sum_{k=1}^{K} y_k = 1, \\
y_k \geq 0, \ k = 1, \ldots, K,
\]

where \( \bar{x}^k \) are the extreme points of \( \mathcal{X} \) and \( K \) is the number of its extreme points. If \( \mathbf{x}^* \) is an optimal solution to \( \text{MP1} \), the optimal solution to \( \textbf{P1} \) is calculated as \( \mathbf{x}^* = \sum_{k=1}^{K} y_k^* \bar{x}^k \). Recall that since \( \mathcal{X} \subset \mathbb{R}^n \), by Caratheodory’s theorem, at most \( n + 1 \) extreme points are needed to represent the optimum point \( \mathbf{x}^* \).

If \( \mathcal{X} \) is defined as the Cartesian product of polytopes, as in the case of the problem \( \textbf{CPP} \), the above transformation still applies. However, another possibility is to apply Caratheodory’s theorem to each polytope separately.
In such a case, CPP can be restated as follows:

\[
(MP2) \quad \min_y f\left(\sum_{j=1}^{K_1} y_{1j} \bar{x}^j_1, \ldots, \sum_{j=1}^{K_m} y_{mj} \bar{x}^j_m\right),
\]

\[
\text{s.t. } \sum_{j=1}^{K_i} y_{ij} = 1, \quad i = 1, \ldots, m,
\]

\[
y_{ij} \geq 0, \quad j = 1, \ldots, K_i, \quad i = 1, \ldots, m,
\]

where \(\bar{x}^j_i\) are the \(K_i\) in total extreme points of the \(i^{th}\) polytope \(\mathcal{X}_i\) in the Cartesian product \(\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_m\). Clearly, MP2 includes a convexity constraint for each polytope in the product and is therefore itself a minimization problem over a Cartesian product of simplices.

The simplicial decomposition approach is based on solving a restricted version of MP1 or MP2, called the restricted master problem. The restriction concerns the number of extreme points used, that is, a number \(\bar{K}\) of extreme points much lesser than the actual number \(K\) is used. If the produced solution is not optimal for the original problem then new extreme points are added and the new restricted problem is solved again. In order to generate the needed extreme point, the Frank-Wolfe subproblems, FW-SUB\(^4\) or (92), are solved in each iteration. Thus, simplicial decomposition can be thought of as an improved version of the original Frank-Wolfe algorithm, where the line search has been replaced by a restricted master of type MP1 or MP2, or as a column generation scheme, where columns (i.e., extreme points) are generated as needed. The algorithm has been proposed and analyzed in its general setting as well as in traffic and flow assignment problems by several authors (e.g. [86, 36, 155, 79, 80, 104]) and is shown to be finite by [79], however, the solution of the restricted master problems does require the application of, in general, infinitely converging algorithms, several of which have been proposed in the literature, including projected second-order methods such as Newton methods [16, 18, 19], reduced gradient methods [104], etc.

The application of simplicial decomposition to the flow problems of Section 3 provides several different opportunities; the master problem can be of the form MP1, i.e., having only one convexity constraint incorporating all origin-destination pairs, or of the form MP2 in which there is one convexity constraint for each origin-destination pair. The latter case is known as the disaggregate simplicial decomposition (DSD), a term coined by [104].
In between there are some other possibilities, such as to have one convexity constraint per origin or per destination, or different other levels of aggregation. It is observed in [104] that the disaggregate case is the more efficient and offers several advantages in the case of traffic assignment problems in road networks. The extreme points generated by the Frank-Wolfe subproblems correspond again to route flows, that is the subproblems are still of the type “all-or-nothing assignment”. Cantor and Gerla [36] seems to be first in applying the simplicial decomposition algorithm to optimal routing in packet-switched networks.

Similar column generation approaches to the flow problems of Section 3 have been proposed in the case of traffic assignment in road networks already by Dafermos [44, 109]. The method is based on solving a restricted version of the path flow formulations FTAP or SFTAP, that is, similarly to the simplicial decomposition approach, “all-or-nothing assignment” problems are solved in order to generate path flows to be included in (40)-(43). The differences from the simplicial approach are therefore basically conceptual rather than substantial [49]. Efficient implementation of the method in this setting are offered by Bertsekas et al [16, 18, 19, 20], where the Goldstein-Levitin-Polyak gradient projection method [15, 110, 17] and various of its extensions incorporating second order information are used in solving the restricted master problem, i.e., the restricted versions of FTAP and SFTAP.

In a hybrid approach (see §4.4 below), [48, 90] solve the restricted master problems applying a regularized Frank-Wolfe approach of Section 4.2. A distributed implementation of the algorithm is realized in [48] (see §4.5 below), while a corresponding sequential, very efficient implementation is presented in [90], where the computational results show that the overall approach is about 50% more efficient when regularized Frank-Wolfe is used to solve the master problem instead of Wolfe’s reduced gradient algorithm [160, 6] employed by [104]. A different hybridization is realized by [106] where, within the frame of a non-linear column generation scheme, the simplicial decomposition subproblems are regularized Frank-Wolfe subproblems, while the master problem is also solved by a regularized Frank-Wolfe algorithm.

### 4.4 Hybridization of Simplicial Decomposition and Frank-Wolfe Regularization

Consider problem (MP2) in the following form
\[(MP) \quad \min_{y \in \mathcal{Y}} f(y),\]
\[
\text{s.t.} \quad y = [y_1, y_2, \ldots, y_m],
\]
\[
y_i \in \mathcal{Y}_i = \left\{ y_i \left| \sum_{j=1}^{K_i} y_{ij} = 1, y_{ij} \geq 0 \right. \right\}, \quad i = 1, \ldots, m, \quad \sum_{i=1}^{m} K_i = n,
\]

where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is continuously differentiable and non-separable and \( \mathcal{Y} = \prod_{i=1}^{m} \mathcal{Y}_i \) is the Cartesian product of the \( m \) simplices. These conditions are also sufficient to guarantee the existence of a solution to \((MP)\).

Considering the form of \((MP)\), we are going to develop a decomposition technique that splits it into \( m \) subproblems, one for each simplex. One straightforward approach that can accomplish this task would be the complete linearization of the objective in the sense of the Frank and Wolfe [57]. Indeed, for any given feasible solution \( y_k = [y_{k1}, \ldots, y_{km}] \), let \( c_i = \nabla_i f(y_k), \ i = 1, \ldots, m, \) denote the gradient of \( f \) with respect to \( y_i \) evaluated at \( y_k \). Then \( P \) is approximated at \( y_k \) by \( m \) linear subproblems:

\[
(FW) \quad \min \sum_{j=1}^{K_i} c_{ij} y_{ij}
\]
\[
\text{s.t.} \quad \sum_{j=1}^{n_i} y_{ij} = 1
\]
\[
y_{ij} \geq 0, \quad j = 1, \ldots, n_i.
\]

Let \( \tilde{y}_i \) be the solution to each \( FW \), and let \( \tilde{y} = [\tilde{y}_1, \ldots, \tilde{y}_m] \). A search direction \( d_k = \tilde{y} - y^k \) is defined and a new feasible solution \( y^{k+1} = y^k + \alpha_k d^k \) is generated by finding \( \alpha_k \in [0, 1] \) that minimizes \( f(y^k + \alpha d^k) \).

The complete linearization approach is simple, however, as already mentioned, it has serious drawbacks: Its rate of convergence is sublinear and jamming occurs. Moreover, for the particular case studied here, the subproblems \( FW \) produce quite meaningless solutions with respect to \( MP \). Indeed, assuming that \( c_{i1} \leq c_{i2} \leq \ldots \leq c_{in} \), the optimal solution of each \( FW \) is \( y_{i1} = 1, y_{ij} = 0, j = 2, \ldots, n_i \). Note that the solution is known as soon as the least value coefficient is known. This value can clearly be identified in \( O(n_i) \) time.

One attractive feature of the approach is that in case of convex \( f \), the subproblem solutions can be used in constructing a lower bound on \( f(y^*) \),
i.e., $f(y^*) \geq \text{LBD}^k \equiv f(y^k) + \nabla f(y^k)^T d^k$. Note that the lower bound can be computed in time $\sum_{i=1}^{m} O(K_i) = O(n)$. This furnishes a rigorous stopping rule, i.e., the iterations are terminated whenever $f(y^k)$ is close to the best known lower bound. However, in view of the easiness by which such lower bounds can be calculated for $\text{MP}$, this stopping rule can be utilized in any other decomposition algorithm that does not suffer from the same drawbacks as the complete linearization. Consider now such a decomposition technique that resembles that of complete linearization, however, the direction generating subproblems are made nonlinear by utilizing the concept of regularization:

$$
\text{(RFWP)} \quad \min \sum_{j=1}^{K_i} c_{ij}y_{ij} + \phi_i(y_i, y_i^k)
$$

s.t.

$$
\sum_{j=1}^{K_i} y_{ij} = 1
$$

$$
y_{ij} \geq 0, \quad j = 1, \ldots, n_i.
$$

The introduction of a regularization term in the objective function of the subproblem is aimed at restricting the distance between the current iterate and the subproblem solution. Retaining nonlinearity in the subproblems has the effect of avoiding the tailing-off phenomena, inherent in the complete linearization and caused by the extreme point subproblem solutions, and the effect of scaling the Frank-Wolfe direction. Moreover, it produces more meaningful subproblem solutions than $\text{FW}$, and the optimal solution to $P$ can, in fact, be obtained from the subproblems. In this application only quadratic regularizations, i.e., $\phi_i(y, y^k) = \frac{1}{2}(y-y^k)^T D_i (y-y^k)$, with positive definite matrices $D_i$, will be considered. A particular choice of $D_i$ will be discussed below.

From Lemma 1, Lemma 2, Theorem 17, and the problem properties we have the following results:

**Corollary 1** If $y^k = [y_1^k, \ldots, y_m^k]$ solves the corresponding $m$ subproblems RFWP, then $y^k$ is also a solution to $P$. The converse is also true.

**Corollary 2** For feasible $y^k = [y_1^k, \ldots, y_m^k]$, let $\tilde{y}^k = [\tilde{y}_1^k, \ldots, \tilde{y}_m^k]$ solve the corresponding $m$ subproblems RFWP, and assume that $y^k \neq \tilde{y}^k$. Then $d^k = [d_1^k, \ldots, d_m^k] = \tilde{y}^k - y^k$ is a feasible direction of descent of $f$ at $y^k$.
Corollary 3  For given feasible $y^k$ and any $i = 1, \ldots, m$, RFWP is feasible and its optimal solution is unique.

Lemma 3  Let $\mu_i$ be the least eigenvalue of $D_i$. Then $\nabla_i f(y^k)^T d_i^k \leq -\mu_i \|d_i^k\|$.  

Proof:  By the positive definiteness of $D_i$, $d_i^T D_i d_i > 0, \forall d_i \in \mathbb{R}^K$, $d_i^T d_i \geq \mu \|d_i\|^2$ and $\mu_i > 0$. Consequently the objective function in RFWP is strongly convex. From the proof of Theorem 2, it follows that $\nabla_i f(y^k)^T d_i^k \leq (y_i^k - \hat{y}_i^k)^T D_i (\hat{y}_i^k - y_i^k)$. Strong convexity implies the result. ■

Lemma 4  The maximum feasible step length is 1.

Proof:  Since $\hat{y}^k$ is feasible, it is always allowed to move from $y^k$ to $\hat{y}^k$. Thus, the step length 1 is feasible. Moreover, in order to guarantee nonnegativeness, the maximum step length should not exceed $\min \{-y_{ij}^k/d_{ij}^k | d_{ij}^k < 0\} \leq 1$. ■

The conceptual algorithm is stated below. Note that the lower bounds, LBD$^k$, are valid only if $f$ is a convex function. The termination criterion that is based on the difference of upper and lower bound can be made inactive by suitable choice of $\epsilon$.

1. algorithm regularized_fw
2. Initialize $y^0 \in \mathcal{Y}$, LBD$^{-1}$ ← $-\infty$, $k$ ← 0
3. do
4. UBD$^k$ ← $f(y^k)$
5. Solve all FW and compute LBD$^k$
6. Let $\hat{y}^k = [\hat{y}_1^k, \ldots, \hat{y}_m^k]$ be the unique solutions to all RFWP
7. $d^k$ ← $\hat{y}^k - y^k$
8. if ($d^k = 0$) exit
9. LBD$^k$ ← max{LBD$^{k-1}$, LBD$^k$}
10. if( (UBD$^k$ − LBD$^k$)/LBD$^k$ ≤ $\epsilon$ ) exit
11. $\alpha_k$ ← arg min $\alpha \in [0,1] f(y^k + \alpha d^k)$
12. $y^{k+1}$ ← $y^k + \alpha_k d^k$
13. $k$ ← $k + 1$
14. end do
15. end regularized_fw
In Theorem 17 we demonstrate under the assumption of exact line searches the global convergence of the approach. In the analysis we ignored the lower bound termination rule. Here, we demonstrate the convergence of the \texttt{regularized_fw} algorithm in the case of inexact line searches in the sense of Armijo. For this purpose we shall make the additional assumption that the gradient of $f$ is Lipschitz continuous, i.e., there exists a constant $\lambda > 0$ such that $\| \nabla f(y^1) - \nabla f(y^2) \| \leq \lambda \| y^1 - y^2 \|$, $\forall y^1, y^2 \in \mathcal{Y}$.

Consider the case of replacing the exact line search with an Armijo step length rule, where $\alpha_i$ in the algorithm is taken to be $\beta^{-i_0}$ and $i_0$ is that of the integers $i = 0, \ldots$, which is the first to satisfy the inequality $f(y^{k+1}) = f(y^k) + \beta^{-i_0}d^k \leq f(y^k) + \gamma \beta^{-i_0} \nabla f(y^k)^T d^k$, where $\gamma, \beta > 0$. In [142], for instance, $\gamma = 0.5$ and $\beta = 2$ are chosen.

**Theorem 18** Under the additional assumption of Lipschitz continuity of the gradient, the \texttt{regularized_fw} algorithm with Armijo steps converges globally to an optimal solution of MP.

**Proof:** We demonstrate first that $i_0$ is finite. Let $w^k = y^k + \xi(\hat{y}^k - y^k)$ for $\xi \in [0, 1]$. By Taylor's formula and the Lipschitzian continuity of the gradient we obtain, for $\alpha \in [0, 1]$, $(f(y^k + \alpha(\hat{y}^k - y^k)) - f(y^k)) = \int_0^\alpha \nabla f(w^k)^T (\hat{y}^k - y^k)d\xi \leq \alpha \nabla f(y^k)^T (\hat{y}^k - y^k) + \int_0^\alpha \lambda \xi \| \hat{y}^k - y^k \|^2 d\xi = \alpha \nabla f(y^k)^T (\hat{y}^k - y^k) + 0.5\alpha^2 \lambda \| \hat{y}^k - y^k \|^2$.

Thus, for $\alpha \leq 2(\gamma - 1) \frac{\nabla f(y^k)^T (\hat{y}^k - y^k)}{\lambda \| \hat{y}^k - y^k \|^2}$, we obtain the inequality $f(y^k) + \alpha d^k - f(y^k) \leq \gamma \alpha \nabla f(y^k)^T d^k$. Since $i_0$ is the first index satisfying the Armijo step length rule, replacing $\alpha$ above by $\beta^{-i}$, we must have $\beta \alpha_k = \beta^{-(i_0 - 1)} > 2(\gamma - 1) \frac{\nabla f(y^k)^T (\hat{y}^k - y^k)}{\lambda \| \hat{y}^k - y^k \|^2}$, and hence $\alpha_k > \frac{2(\gamma - 1) \nabla f(y^k)^T (\hat{y}^k - y^k)}{\lambda \| \hat{y}^k - y^k \|^2}$. Thus, if $d^k$ is a feasible direction of descent, the Armijo inequality will be satisfied after a finite number of trials and the chosen step will satisfy the last inequality.

Convergence to the optimal solution is proven next. Since $\| y \| \leq m, \forall y \in \mathcal{Y}$, is $\| \hat{y}^k - y^k \| \leq 2m, \forall k$. Hence, $\alpha_k > \frac{2(\gamma - 1) \nabla f(y^k)^T (\hat{y}^k - y^k)}{4m^2 \lambda}$ and consequently, by construction, $f(y^{k+1}) - f(y^k) \leq \frac{2(\gamma - 1) \| \nabla f(y^k)^T (\hat{y}^k - y^k) \|}{4m^2 \lambda}$. Adding the latter for $k = 0, 1, \ldots, q - 1$ we obtain $f(y^q) - f(y^0) \leq \frac{2q(\gamma - 1)}{4m^2 \lambda} \sum_{k=0}^{q-1} \| \nabla f(y^k)^T d^k \|^2$.

Since $f(y)$ is bounded from below in $\mathcal{Y}$ by $f(y^*)$, the minimum value of $f$ on $\mathcal{Y}$, we have $\sum_{k=0}^{q-1} \| \nabla f(y^k)^T d^k \|^2 \leq \frac{4m^2 \beta \lambda}{2(\gamma - 1)} [f(y^*) - f(y^q)] \leq \frac{4m^2 \beta \lambda}{2(\gamma - 1)} [f(y^*) - f(y^*)]$. Hence, the series $\sum_{k=0}^{\infty} \| \nabla f(y^k)^T d^k \|^2$ converges. But this is possible only if $\nabla f(y^k)^T d^k \to 0$. 53
Thus, as in the proof of Theorem 17, there are subsequences \( \{y^k\}_{k \in M} \rightarrow y^\star \) and \( \{\hat{y}^k\}_{k \in M} \rightarrow \hat{y} \) such that \( \nabla f(y^\star)^T(\hat{y} - y^\star) = 0 \). By using the same argumentation as in the proof of Theorem 6, the desired result is obtained.

\[ \blacksquare \]

### 4.4.1 Polynomi ally bounded dual algorithm for the solution of the subproblems

We assume here that at iteration \( k \), the regularization functions \( \phi_i(y_i, y^k_i) = \frac{1}{2}(y_i - y^k_i)^T D_i(y_i - y^k_i) \) with diagonal \( D_i \), e.g., \( D_i = \text{diag}(\nabla^2 f(y^k)) \), perturbed if necessary to positiveness, are chosen for each \( i \in \{1, \ldots, m\} \). Dropping constant terms and indexing, each subproblem is thus of the form:

\[
\text{(QP)} \quad \min \sum_{j=1}^n \frac{1}{2} d_j y_j^2 - l_j y_j \\
\text{s.t.} \quad \sum_{j=1}^n y_j = 1 \\
0 \leq y_j \leq 1, \ j = 1, \ldots, n.
\]

Problems similar to QP arise as subproblems in a variety of applications and have therefore attracted considerable attention. In [81], an \( O(n \log n) \) algorithm was derived for solving QP with arbitraty upper bounds on the variables. Subsequently, a \( O(n) \) algorithm was derived in [33] for solving QP with arbitrary lower and upper bounds on the variables, and arbitrary coefficients and right-hand side in the equality constraint. In [133] the latter algorithm is modified to an \( O(n^2) \) worst case complexity but linear average complexity. Although the different authors based their algorithmic development on different concepts, ranging from Karush-Kuhn-Tucker optimality conditions [133] to parametric programming [33], the fundamental ideas are common and can be unified in terms of Lagrangian duality. Let \( \lambda \in \mathbb{R} \) be the Lagrangian multiplier associated with the equality constraint and consider the Lagrangian inner subproblem:

\[
[\text{QP}(\lambda)] \quad \min \sum_{j=1}^n \frac{1}{2} d_j y_j^2 + (\lambda - l)_j y_j \\
\text{s.t.} \quad 0 \leq y_j \leq 1, \ j = 1, \ldots, n.
\]

Since \( d_j > 0, \forall j \), \( \text{QP}(\lambda) \) has a unique finite solution \( y(\lambda) \) defined by

\[
y_j(\lambda) = \begin{cases} 
0, & \text{if } (l_j - \lambda)/d_j \leq 0 \\
1, & \text{if } (l_j - \lambda)/d_j \geq 1 \\
(l_j - \lambda)/d_j, & \text{otherwise},
\end{cases}
\]

54
Thus, the Lagrangian dual to \( QP \) is

\[
\text{[QPLD]} \quad \max -\lambda + \sum_{j=1}^{n} \frac{1}{2} d_j y_j(\lambda)^2 + (\lambda - l_j) y_j(\lambda).
\]

Problem \( QP \) is always feasible since, trivially, \( y_j = \frac{1}{n}, \ n = 1, \ldots, n \), is a feasible solution in the interior of the feasible region. Its optimal solution is therefore finite. Thus, by [6, Theorem 6.2.4], \( \text{QPLD} \) has a finite optimum \( \lambda^* \), and the optimal objective function values of \( QP \) and \( \text{QPLD} \) are equal. Moreover, due to the uniqueness of \( y(\lambda) \) in \( \text{QP}(\lambda) \), the dual function is differentiable [6, Theorem 6.3.3] with \( \sum_{j=1}^{n} y_j(\lambda) - 1 \) being its derivative. Hence, by the concavity of the dual function [6, Theorem 6.3.1], the derivative is monotone nonincreasing [6, Theorem 3.3.4], continuous and, by the definition of \( y(\lambda) \), piecewise linear (see Figure 4). Then \( \lambda^* \) is the solution of the equation \( \sum_{j=1}^{n} y_j(\lambda) = 1 \), and \( y(\lambda^*) \) solves \( QP \). The break-points (i.e., the points where the linear pieces change) are defined by \( \alpha_j = l_j - d_j \) and \( \beta_j = l_j, \ \forall j \). These points can be searched in order to identify the linear piece \([\alpha, \beta]\) for which \( \sum_{j=1}^{n} y_j(\lambda) = 1 \), i.e., \( \lambda^* \in [\alpha, \beta] \). The mentioned algorithms differ with respect to how this search is performed [33, 81, 133]. More specifically, the algorithms calculate the \( 2n \) break points \( \alpha_j \) and \( \beta_j \) which for simplicity are denoted by \( b_k, 1 \leq k \leq 2n \).

The algorithm of [81] ranks the break points in ascending sequence: \( b_1 \leq b_2 \leq \cdots \leq b_k \leq \cdots \leq b_{2n} \). Thereafter, it attempts a binary search in the initial interval \([a_0, b_0] \equiv [b_1, b_{2n}]\). The mean of the initial interval \( a_0, b_0 \) is considered to be the median break point \( b_m \). The search is then repeated in the interval \( [a_1, b_1] \) which is equal either to the interval \([b_1, b_m] \) or to the interval \([b_m, b_{2n}] \) depending on the value of \( g(b_m) \). The process continues until either \( g(b_m) = 0 \) or the interval contains only one line segment. Thus, the desired \( \lambda^* \) coincides with the median \( b_m \) in the first case and is calculated through linear interpolation in the second case. The algorithm proposed by [33] avoids ranking of the break points in ascending sequence. In contrast, it is based on the calculation of two medians, one for break points \( \alpha_j \) and one for break points \( \beta_j \). Seeing that the computation complexity of the median is lower than that of the ranking break points and since the sets of the break points \( \alpha_j \) and \( \beta_j \) decline by at least at \( 1/4 \) in each iteration, Brucker, [33] proved that the algorithm has complexity \( O(n) \) instead of \( O(n \log n) \) which was proposed by [81]. Pardalos and Kovoor ([133] employed a randomized form of Brucker’s algorithm. Their algorithm uses a fast approximate computation of the median instead of its precise estimation. The approximation is computed through random selection on time \( \Theta(1) \). While the worst case
complexity increases to $\Theta(n^2)$ the complexity of the algorithm is on average $o(n)$ . Empirical tests for $n = 500$ and 4000 demonstrate the superiority of their algorithm over the algorithm of [81] by a factor of 5.

It is possible to compute the $\lambda^*$ without calculating the break points and without making use of discrete algorithms. The bisection algorithm of the nonlinear programming is able to identify the $\lambda^*$ starting from an initial interval $[a_0, b_0]$ with $g(a_0)g(b_0) < 0$. Instead of the median, the midpoint of $\frac{(a_0 + b_0)}{2}$ bisecting the $[a_0, b_0]$ is computed and tested. At each iteration, the interval is reduced by half. For a desired reduction of the initial interval of length $\varepsilon$, the maximum number of iterations required is $\log_2 \frac{b_0 - a_0}{\varepsilon}$. The final $\lambda^*$ is obtained by linear interpolation on the final interval

$$\lambda^* = a_k + \frac{(b_k - a_k)(1 - g(a_k))}{g(b_k) - g(a_k)}.$$ (116)

Figure 4: The derivative of the objective function in QPLD: Breakpoints and the optimal multiplier.
4.4.2 Implementation and Computational Results for the Stochastic Transportation Problem

The application of the simplicial decomposition method to STP results in column generation sub-problems of the form:

\[
\text{(CGS)} \quad \min \sum_{i=1}^{m} \sum_{j=1}^{n} \bar{c}_{ij}^{(k)} x_{ij} \tag{117}
\]

s.t \quad \sum_{j=1}^{n} x_{ij} \leq q_i, \; \forall i \tag{118}

\quad x_{ij} \geq 0 \; \forall i, \; \forall j \tag{119}

where

\[
\bar{c}_{ij}^{(k)} = \frac{\partial f(x^{(k)}, r^{(k)})}{\partial x_{ij}} + \frac{\partial f(x^{(k)}, r^{(k)})}{\partial r_j}, \; \forall i, \; \forall j \tag{120}
\]

with

\[
r_j^{(k)} = \sum_{i=1}^{m} x_{ij}^{(k)}, \; \forall j \tag{121}
\]

and \( f(x, r) \) and denotes the objective function of the initial problem.

Noticeably, the CGS is separated into \( m \) linear knapsack problems which are easy from computational point of view to solve. This property is used by Cooper and LeBlanc \[43\] when implementing an algorithm based on the Frank Wolfe to solve the STP.

Concerning the choice of method for solving the quadratic knapsack sub-problems, we performed an extensive number of tests with random data in order to select the final solution algorithm. Some of these tests are presented in Table 3. All of the algorithms were implemented in Fortran 95 and compiled with Lahey/Fujitsu lf95 version 6.2 compiler under SuSE Linux 9.1, and executed on a Pentium IV at 3.02 GHz.

As shown in Table 3 the discrete algorithms are not faster than the continuous bisection process. This difference becomes more intense as the size of problems increases. Among the discrete algorithms, the algorithm of Pardalos and Kovoor [133] is faster than that of Helgason et al. [81] which is slightly faster than of Brucker [33] although the theoretical complexity of the
Table 3: Comparison of the runtime of the algorithms

<table>
<thead>
<tr>
<th>Problem size</th>
<th>Execution time (secs)</th>
<th>Helgason</th>
<th>Brucker</th>
<th>Pardalos</th>
<th>Bisection $\varepsilon = 0.00001$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>5.99E-3</td>
<td>3.000E-3</td>
<td>7.998E-3</td>
<td>1.000E-3</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>3.99E-2</td>
<td>4.099E-2</td>
<td>2.599E-2</td>
<td>2.000E-3</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>0.112</td>
<td>9.69E-2</td>
<td>7.099E-2</td>
<td>3.999E-3</td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>0.257</td>
<td>0.263</td>
<td>0.178</td>
<td>1.199E-2</td>
<td></td>
</tr>
<tr>
<td>7000</td>
<td>0.390</td>
<td>0.398</td>
<td>0.253</td>
<td>1.500E-2</td>
<td></td>
</tr>
<tr>
<td>10000</td>
<td>0.619</td>
<td>0.628</td>
<td>0.312</td>
<td>2.200E-2</td>
<td></td>
</tr>
</tbody>
</table>

latter is log $n$ better than that of the algorithm by Helgason et al. [81]. It should be underlined that increasing somewhat $\varepsilon$ in the continuous bisection method the practical accuracy of the calculation of $\lambda^*$ is slightly reduced, while the performance of the algorithm is accelerated considerably.

The data used to derive the computational results presented in this section were created randomly and in accordance with the basic principles proposed Cooper and LeBlanc [43] and LeBlanc et al. [108] respectively. The first principle creates problems of 200 destinations and 5 to 25 sources. The second principle creates problems with 100 destinations and 5 to 25 sources. For a problem of size $25 \times 100$ which was created according to the first principle, about 96% of the optimal objective value is arisen by the non-linear terms of shortage and holding costs. For problems of size $25 \times 100$ this percentage is about 60%.

The algorithm was implemented in Fortran 95 and compiled with Lapey/Fujitsu if95 version 6.2 compiler under SuSE 9.1, and executed on a Pentium IV at 3.02 GHz. Figure 5 represents the average of the maximum sizes of the master problems over the relative accuracy required for their solution.

The computational time required to solve the testing problems with relative accuracy test is shown in Figure 6.

Finally, Figure 7, shows the computational time required to solve the mas-
Figure 5: Size of restricted master problem in columns compared to relative accuracy

Figure 6: Computational time compared to the required relative accuracy
ter problems created according to the two principles for a range of accuracy between $10^{-4}$ and $10^{-2}$.

Problems created by the principle of Cooper and LeBlanc [43]

![Graph](image1)

Problems created by the principle of LeBlanc et al. [108]

![Graph](image2)

Figure 7: Comparison of the computational results and the problem’s size with accuracy 0.01 %

### 4.4.3 Tailoring the Algorithm to the Traffic Assignment

As already discussed, solving **FTAP** requires repeated solutions of restricted master problems in the form **MP** and more specifically of the following form when using disaggregate representation (see e.g. [104]):
\[(\text{RMP}) \quad \min \sum_{a \in A} f_a \left( \sum_{k \in K} r_k \sum_{p \in \Pi_k} \delta_{kap} \lambda_{pk} \right) \]

s.t. \[\sum_{p \in \Pi_k} \lambda_{pk} = 1, \forall k \in K\]
\[\lambda_{pk} \geq 0, \forall p \in P_k, \forall k \in K\]

where $\lambda_{pk}$ is the fraction of the total flow origin-destination (O-D) demand on path $p_k \in P_k$. If the RMP solution $[\lambda_{pk}]$ does not solve FTAP, where $h_{pk} = r_k \lambda_{pk}, p \in P_k, k \in K$, then the set $P_k$ is augmented by solving the following column generating shortest path subproblems which are separated over $k$ and are obtained by linearizing AFTAP at the current flow $x_a = \sum_{k \in K} \sum_{p \in \Pi_k} \delta_{kap} h_{pk}$ (where $c_a = s_a(x_a)$):

\[[\text{SPP}_k] \quad \min \tilde{f}_k(x) = \sum_{a \in A} c_a x_a^k \]

s.t. \[\sum_{a \in S(i)} x_a^k - \sum_{a \in T(i)} x_a^k = \begin{cases} r_k & \text{if } o(k) = i \\ -r_k & \text{if } d(k) = i \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in \mathcal{N}\]
\[x_a^k \geq 0, \forall a \in A\]

Applying the regularized FW to the RMP, we can compute the QP objective terms as $d_{pk} = r_k^2 \sum_{a \in A} \delta_{kap} s_a'(x_a), p \in P_k, k \in K$, and $l_{pk} = d_{pk} \lambda_{pk} - r_k \sum_{a \in A} \delta_{kap} s_a(x_a), p \in P_k, k \in K$.

4.4.4 The hybrid algorithm for the traffic assignment problem

With the main subproblems stated, we can now give the algorithm for the solution of the traffic assignment problem. Define UBD as the upper bound on the objective, by LBD the lower bound (obtainable from the shortest path problems) and by MLBD the lower bound on the restricted master problem objective. In the description below we use the shorthand notation like $x_a \leftarrow 0$ to mean “assignment for all $a$”, and $\iff$ to mean “obtainable from the solution of”.

**algorithm** DSD-RFW

**Heuristic.** Generate the first path for each O-D pair at zero arc flow,
and assign the full O-D demand to the path, i.e., perform an all-or-nothing assignment.

1. $x_a \leftarrow 0$, $c_a \leftarrow s_a(x_a)$, $\mathcal{P}_k \leftarrow \emptyset$.
2. $\hat{p}_k \leftarrow$ Solve $\text{SPP}_k$, $\lambda_{\hat{p}_k} \leftarrow 1$, $\mathcal{P}_k \leftarrow \mathcal{P}_k \cup \hat{p}_k$.
3. $x_a \leftarrow \sum_{k \in \mathcal{K}} r_k \sum_{p \in \mathcal{P}_k} \delta_{kap} \lambda_{pk}$.
4. $\text{UBD} \leftarrow f(v)$, $\text{LBD} \leftarrow -\infty$.

**Main solver.** Generate paths based on the current arc delays. Augment the set of generated paths. Solve the master problem over this restricted set.

**Column generation.** Shortest path based on current arc delays. Augment the set of generated paths if not previously included. Compute a lower bound on the objective.
5. $c_a \leftarrow s_a(x_a)$.
6. $\{\hat{f}_k(x), \hat{p}_k\} \leftarrow$ Solve $\text{SPP}_k$.
7. If $\hat{p}_k \not\in \mathcal{P}_k$ then $\mathcal{P}_k \leftarrow \mathcal{P}_k \cup \hat{p}_k$, $\lambda_{\hat{p}_k} \leftarrow 0$.
8. $\text{LBD} \leftarrow \max \{\text{LBD}, \text{UBD} + \sum_{k \in \mathcal{K}} \hat{f}_k(x) - \sum_{a \in \mathcal{A}} c_a x_a\}$.

**Convergence test.** Terminate the algorithm if the relative objective error is lower than some a priori set constant.
9. If $|(\text{UBD}-\text{LBD})/\text{LBD}| \leq \epsilon_1$ then terminate.

**Restricted master.** Solve the restricted master defined for the set of generated paths. In each iteration, a $\text{QP}$ is solved for each O-D pair.
10. $\text{MLBD} \leftarrow -\infty$.
11. $d_{pk} \leftarrow r_k^2 \sum_{a \in \mathcal{A}} \delta_{kap} s'_a(x_a)$, $l_{pk} \leftarrow d_{pk} \lambda_{pk} - r_k \sum_{a \in \mathcal{A}} \delta_{kap} s_a(x_a)$.
12. $\text{MLBD} \leftarrow \max [\text{MLBD}, \text{UBD} + \sum_{k \in \mathcal{K}} r_k \min_{p_k \in \mathcal{P}_k} \{\sum_{a \in \mathcal{A}} \delta_{kap} s_a(x_a)\}] - \sum_{a \in \mathcal{A}} c_a x_a$.
13. If $|(\text{UBD}-\text{MLBD})/\text{MLBD}| \leq \epsilon_2$ then the master is sufficiently solved. Goto 5.
14. $\lambda^{old}_{pk} \leftarrow \lambda_{pk}$, $x^{old}_a \leftarrow x_a$.
15. $\lambda_{pk} \leftarrow$ Solve $\text{QP}$.
16. $x_a \leftarrow \sum_{k \in \mathcal{K}} r_k \sum_{p \in \mathcal{P}_k} \delta_{kap} \lambda_{pk}$.
17. $\lambda^{dir}_{pk} \leftarrow \lambda_{pk} - \lambda^{old}_{pk}$, $x^{dir}_a \leftarrow x_a - x^{old}_a$.
18. step $\leftarrow$ Apply Armijo line search.
18’ (or step $\leftarrow$ min\{1, $-\sum_{a \in \mathcal{A}} s'_a(x^{old}_a) x^{dir}_a / \sum_{a \in \mathcal{A}} s'_a(x^{old}_a) (x^{dir}_a)^2\}$)
\[ \lambda_{pk} \leftarrow \lambda_{pk}^{\text{old}} + \text{step} \times \lambda_{\text{dir}}, \quad x_a \leftarrow x_a^{\text{old}} + \text{step} \times x_a^{\text{dir}}. \]

19. \[ UBD \leftarrow f(x). \]
20. \[ \text{Goto 11.} \]

### 4.4.5 Computational Results

We implemented the algorithm in C++ and report here a total of nine tests on three real world networks requesting three levels of relative objective error: (i) the network of the city of Barcelona, Spain, which has 1020 nodes, 2522 arcs and 7922 origin-destination (O-D) pairs, (ii) the network of Linköping, Sweden, which has 335 nodes, 882 arcs and 12372 O-D pairs, and (iii) the network of Winnipeg, Canada, which has 1052 nodes, 2836 arcs and 4344 O-D pairs.

Our code (DSD-RFW) is compared to the Fortran 77 code (DSD) by [104], a C++ version of which has been added as a toolbox [111] to the free software package Scilab [88], in terms of execution time, number of main iterations and number of line searches. The essential difference between the two codes is the restricted master solver, where DSD uses a reduced gradient technique, while our DSD-RFW utilizes the regularized fw algorithm.

We adjusted the parameter \( \gamma \) (with \( \beta = 2 \) constant) in the Armijo line search procedure to obtain the fastest possible running times for both codes. The best \( \gamma \) was always between 0.2 and 0.3. We also tested various C++ and Fortran compilation parameters since this has great impact on the computing speed, resulting in the following switches when using Sun (Oracle) C++ and Fortran compilers: `-Bstatic -dalign -fsimple -O5 -xlibmopt -xlibmil`. The results are provided in Tables 1, 2 and 3.

As can be observed in Tables 1, 2, and 3, the DSD-RFW code is in average a factor 1.7 times faster than the DSD code. The DSD-RFW code uses a much lower number of expensive line searches in order to obtain the requested relative error which is one explanation for the lower CPU times. The main reason for this behavior is that we use second order information in the regularized master subproblem.

We can also see that the number of main iterations are in general higher for the DSD-RFW code, which at a first glance seems strange. However, the explanation is that the two codes use different termination strategies for the master problem: the DSD-RFW code uses a bounding strategy for the master objective, while the DSD code terminates when the relative difference between
Table 4: Statistics on CPU time (in seconds), number of main iterations (#MI), and number of line searches (#LS) for algorithms DSD and DSD-RFW. Relative objective error achieved: 0.5%.

<table>
<thead>
<tr>
<th>Network</th>
<th>Code</th>
<th>CPU</th>
<th>#MI</th>
<th>#LS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barcelona</td>
<td>DSD</td>
<td>61.3</td>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>DSD-RFW</td>
<td>31.3</td>
<td>10</td>
<td>28</td>
</tr>
<tr>
<td>Linköping</td>
<td>DSD</td>
<td>54.2</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DSD-RFW</td>
<td>48.3</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>Winnipeg</td>
<td>DSD</td>
<td>53.8</td>
<td>6</td>
<td>114</td>
</tr>
<tr>
<td></td>
<td>DSD-RFW</td>
<td>30.4</td>
<td>12</td>
<td>35</td>
</tr>
</tbody>
</table>

Table 5: Statistics on CPU time (in seconds), number of main iterations (#MI), and number of line searches (#LS) for algorithms DSD and DSD-RFW. Relative objective error achieved: 0.1%.

<table>
<thead>
<tr>
<th>Network</th>
<th>Code</th>
<th>CPU</th>
<th>#MI</th>
<th>#LS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barcelona</td>
<td>DSD</td>
<td>92.6</td>
<td>6</td>
<td>149</td>
</tr>
<tr>
<td></td>
<td>DSD-RFW</td>
<td>58.8</td>
<td>10</td>
<td>59</td>
</tr>
<tr>
<td>Linköping</td>
<td>DSD</td>
<td>53.8</td>
<td>7</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>DSD-RFW</td>
<td>31.7</td>
<td>6</td>
<td>36</td>
</tr>
<tr>
<td>Winnipeg</td>
<td>DSD</td>
<td>135.5</td>
<td>7</td>
<td>281</td>
</tr>
<tr>
<td></td>
<td>DSD-RFW</td>
<td>59.6</td>
<td>11</td>
<td>81</td>
</tr>
</tbody>
</table>
two successive master objective values is less than some constant which is updated in a predetermined fashion. We suspect that the DSD master is solved too accurately, especially in the later stages of the algorithm, and that the overall performance would be better if the master solver was terminated earlier so that the (overall) lower bound from the shortest path phase could be updated more often. In fact, the results for DSD in Tables 1, 2, and 3 are from a modified code where we terminate the master solver after a fixed number (found through experimentation) of iterations, if the original termination criterion fails to do so in good time. Hence, the DSD code would have performed even worse without this modification.

We also experimented with a master solver where the Armijo line search procedure is replaced with a step based on the quadratic approximation of the objective (step 18 in the pseudocode). Although we do not offer a convergence proof or detailed computational results, this version of DSD-RFW is also consistently faster than the DSD code albeit only about 10% on average. Compared to the DSD-RFW code with Armijo line search, it requires a higher number of main iterations and also a greater number of iterations within the master solver in order to converge. The main advantage with this approach would be the possibility it offers to get rid of the inherently sequential line searches in connection with a parallelization of the DSD-RFW algorithm, which we discuss in the following.
Despite of considerable advances in the development of parallel optimization algorithms [20, 134], no particular attempts had been made for the development of parallel algorithms for the solution of large scale convex cost multi-commodity flow problems. In particular, no other parallel implementation of the simplicial decomposition schemes for the traffic assignment problem had been reported before that of [48]. However, related activities had been documented. For instance, [114] vectorize the linearization algorithm of Frank-Wolfe on a CRAY supercomputer for the traffic assignment problem and report computational advantages for randomly generated networks. An approach similar to [122] is utilized in [37] together with orthogonal projections in order to develop and implement a parallel algorithm for single-commodity, convex cost, bipartite transportation problems. They report computations and comparisons for an implementation on a 16 processor T800 Transputer system. [53] report results for the quadratic cost, bipartite, transportation problem on a Connection Machine CM-5 platform using an alternating direction method of multipliers. It is true, on the other hand, that fine-grained or SIMD parallelism had attracted considerable attention. The work of Zenios and his associates [139, 141, 138, 140] had brought parallel implementations of simplicial decomposition schemes, in connection with the quadratic penalty approach, for the solution of capacitated network flow problems. They report on computational results for large-scale linear problems on the CM-2 platform.

In this subsection we describe a synchronous single program multiple data (SPMD) implementation of the hybridized simplicial decomposition and regularized Frank-Wolfe algorithm as adapted by [48]. There are four main reasons for this choice:

i) the convergence properties are exactly the same as for the sequential algorithm,

ii) it is relatively easy to implement since the compute nodes (processors) are doing (virtually) the same thing,

iii) the FTAP problem structure suits this parallelization model nicely, and

iv) the approach is easily adapted to modern shared-memory multicore systems under the OpenMP parallel computing model [38, 40].
In the following we describe the data structures and distribution of data, communication (message passing) patterns as implemented in [48].

### 4.5.1 Data structures and distribution

By using the TAP\textsubscript{h} model, we can readily see that there are two main data structures:

1. **Network.** The network is defined by the number of nodes, links (with corresponding delay functions) and the underlying graph. The graph is stored as a sparse adjacency list of size $|\mathcal{N}| + |\mathcal{A}|$ and the link data (flow, three delay function terms, and two for temporary calculations) are stored in vectors of size $|\mathcal{A}|$. All compute nodes hold the entire network data so (given that the compute nodes know the present link flow) the link delay can be computed in parallel and the shortest path problems can be solved in parallel without any communication.

2. **OD-pairs.** An OD-pair is defined by its origin, destination and the demand of flow that is to carried between the two. All processors hold this (constant) information (of size $O(|\mathcal{K}|)$ to reduce the amount of data to be communicated if load balancing (see Section 4.5.1.2) is used. Furthermore, in the disaggregated simplicial decomposition case, there are also routes associated with the OD-pairs. For each OD-pair there is a structure holding a linked list of the routes (a dynamically allocated vector of network link indices) which has been generated, i.e., the set $\mathcal{P}_k$ (or $\mathcal{L}_k$) from the subproblem phase $\text{SPP}_k$. The same structure also holds the present route flow. The total size of these structures can be estimated with $O(|\mathcal{K}||\mathcal{A}|)$, since, in general, there are only a handful of routes generated per OD-pair in an user equilibrium solution. This is the (possibly huge set of) data which is to be divided among the compute nodes.

By distributing the OD-pairs over the processors we obtain a data distribution which allows for the communication-less solution of the route generating shortest path problems $\text{SPP}_k$ in parallel. Furthermore, within the master problem $\text{RMP}$ we can solve the quadratic knapsack problems (see §4.4.1) in parallel without communication.

By summation over all routes in $\mathcal{P}_k$ and subsequently over $\mathcal{K}$, we obtain the total flow on every network link and, thus, we can compute the objective
and the link gradients, i.e., the link cost or link delay (c.f. §4.4.3). This is the obvious drawback of the data distribution chosen since we need to collect the link flow from the compute nodes every time we evaluate the objective and the link delay. This procedure is performed once per iteration in the master problem solver and once per main iteration (see Section 4.5.2). Furthermore, the amount to be communicated will be linearly dependent on the number of parallel processors used, since they hold their share of the link flow on all links. It is virtually impossible to arrange the OD-pair distribution so that we can guarantee that the routes generated will only use a certain set of the links, hence, all compute nodes must in all practical cases have all link information.

4.5.1.1 Communication (message passing)
Only one type (essentially) of communication is used in the proposed algorithm. Denote it reduce-add and multicast for further reference.

It works as follows: all nodes compute their share of the data in question. A ‘reduction with add’ operation is then performed to gather and compute (summation) the total result, which in its turn is sent (multicast) back to all compute nodes. Optimal algorithms and their time complexity for these operations can be found in, e.g., [20, Section 1.3].

4.5.1.2 Load balancing
In order to fully utilize the processing power it is essential that the compute nodes use equal amount of time to process their share. This is especially true for our synchronous algorithm. Since the workload changes dynamically (and we can not easily know how much beforehand) iteration by iteration, there is a strong possibility that the computing time for each processor will differ significantly after a few iterations if no load balancing is performed. A balancing scheme was therefore incorporated into the code. It performs local load balancing between pairs of processors by comparing their compute time and transferring OD-pairs, i.e., the structure holding the route data (see Section 4.5.1, item 2) in order to attempt to equalize the compute time (see Figure 8). Denote this procedure balance-load for further reference. Computational tests has proven this to be efficient, especially if the algorithm is used for solving several snapshots of the network with varying OD-pair demand for each snapshot. One snapshot is in general solved with only a handful major iterations, so there is little to gain in this case.
Figure 8: Local load balancing scheme.

4.5.1.3 Synchronization

The reduce-add and multicast and balance-load routines are implemented so that the processors wait for the incoming data before continuing (i.e., blocking receives). This forces the processors to operate synchronously. Hence, at all times they will know the same information as a single processor would (running the sequential version of the algorithm). This ensures that the parallel algorithm has exactly the same convergence properties as the sequential algorithm (c.f. [90]).

4.5.2 The distributed disaggregate simplicial decomposition algorithm

In this section we provide a pseudo code for the proposed parallel algorithm.

Let \( C \) denote the set of compute nodes and let \( \mathcal{K}^c \) be the set of OD-pairs that compute node \( c \in C \) deals with. Clearly \( \cup_{c \in C} \mathcal{K}^c = \mathcal{K}, \mathcal{K}^i \cap \mathcal{K}^j = \emptyset, \forall i, j \in C, i \neq j \). Let \( x^c \) denote the vector of link flows \([x^c_a]\), where \( x^c_a = \sum_{k \in \mathcal{K}^c} \sum_{p_k \in P_k} h_{pk} \delta_{kap} \), i.e., the part of the link flow that the generated routes held by compute node \( c \) carry. Let \( \hat{f}^c(x) = \sum_{k \in \mathcal{K}^c} \hat{f}_k(x) \) be the total shortest path cost for problems \( \text{SPP}_k \) which are solved by processor \( c \). Finally, define UBD as the upper bound on the objective and by LBD the lower bound.

Note that all processors run the same program, but each have different data sets (which correspond to \( \mathcal{K}^c \)), i.e., it is a SPMD algorithm.

4.5.2.1 Algorithm SPMD-DDSD

**Initialization.** Data input and initial distribution of OD-pairs. The \(|\mathcal{K}|\) OD-pairs are evenly distributed over the \(|C|\) compute nodes.
Heuristic. Generate the first route for each OD-pair at zero link flow, and
assign the full OD demand to the route — an all-or-nothing assignment.

1. \( \mathbf{x} \leftarrow 0 \), \( c_a \leftarrow s_a(\mathbf{x}) \), \( \mathcal{P}_k \leftarrow \emptyset \).
2. \( \hat{\mathbf{p}}_k \leftarrow \text{Solve } \text{SPP}_k(c_a, \ldots) \), \( h_{\mathbf{pk}} \leftarrow r_k \), \( \mathcal{P}_k \leftarrow \mathcal{P}_k \cup \hat{\mathbf{p}}_k \).
3. \( \mathbf{x} \leftarrow \text{reduce-add_and_multicast}(\mathbf{x}^c) \)
4. UBD \( \leftarrow f(\mathbf{x}) \), LBD \( \leftarrow -\infty \)

Main solver. Generate routes based on the current link delays. Augment
the set of generated routes. Solve the master problem over the restricted
set of generated routes.

Subproblem solver. Shortest path based on current link delays. Aug-
ment the set of generated routes if not previously included. Compute a lower bound on the objective.

5. \( c_a \leftarrow s_a(\mathbf{x}) \).
6. \( \{ \hat{f}_k(\mathbf{x}), \hat{\mathbf{p}}_k \} \leftarrow \text{Solve } \text{SPP}_k(c_a, \ldots) \)
   If \( \hat{\mathbf{p}}_k \notin \mathcal{P}_k \) then \( \mathcal{P}_k \leftarrow \mathcal{P}_k \cup \hat{\mathbf{p}}_k \), \( h_{\mathbf{pk}} \leftarrow 0 \).
7. \( \hat{f}(\mathbf{x}) \leftarrow \text{reduce-add_and_multicast}(\hat{f}^c(\mathbf{x})) \)
8. LBD \( \leftarrow \max\{ \text{LBD}, \text{UBD} + \hat{f}(\mathbf{x}) - \sum_{a \in \mathcal{A}} c_a x_a \} \)

Convergence test. Terminate algorithm if the relative objective error
is below some a priori set constant.

9. If \( (\text{UBD} - \text{LBD})/\text{LBD} \leq \varepsilon \) then Terminate!

Restricted master solver. Solve the equilibrium problem over the
restricted set of generated routes. Each iteration the objective is
approximated with a separable (over the OD-pairs) quadratic func-
tion; see §4.4.1.

10. \( d_{pk} \leftarrow \sum_{a \in \mathcal{A}} \delta_{kap} s'_a(x_a), \quad l_{pk} = \sum_{a \in \mathcal{A}} \delta_{kap} s_a(x_a) \).
11. \( h^\text{old}_{pk} \leftarrow h_{pk} \), \( x^\text{old}_a \leftarrow x_a \)
12. \( h_{pk} \leftarrow \text{Solve } \text{QPP}_k(h^\text{old}_{pk}, \ldots) \)
13. \( \mathbf{x} \leftarrow \text{reduce-add_and_multicast}(\mathbf{x}^c) \)
14. \( h^\text{dir}_{pk} \leftarrow h_{pk} - h^\text{old}_{pk} \), \( x^\text{dir}_a \leftarrow x_a - x^\text{old}_a \)
15. \( \text{step} \leftarrow \min\{ 1, -\sum_{a \in \mathcal{A}} s_a(x^\text{old}_a)(x^\text{dir}_a) / \sum_{a \in \mathcal{A}} s'_a(x^\text{old}_a)(x^\text{dir}_a)^2 \} \)
16. \( h_{pk} \leftarrow h_{pk}^{\text{old}} + \text{step} \times h_{pk}^{\text{dir}}, \quad x_a \leftarrow x_a^{\text{old}} + \text{step} \times x_a^{\text{dir}} \)
17. \( \text{UBD} \leftarrow f(x) \)
18. Terminate master after (a priori set) number of iterations. Return new equilibrium flow.

**Load equalization.** Re-distribute OD-pair data to obtain equal running times for the compute nodes.

20. balance-load

### 4.5.3 Implementation and Computational Results

The algorithm was written in C++ using the GNU C++ compiler [154]. For message passing purposes we used the Parallel Virtual Machine (PVM) library [67]. The two main embedded routines needed are a shortest path and a quadratic knapsack solver. As a shortest path solver we implemented the L-THRESHOLD code in [64]. The quadratic knapsack solver is an implementation of the \( O(n \log n) \) time code in [81].

To evaluate the implementation of the proposed algorithm we ran some numerical tests on two parallel platforms and three relatively large scale real world networks.

#### 4.5.3.1 Network descriptions

We have tested the proposed algorithm on three relatively large scale real world traffic networks.

**4.5.3.1.1 The Barcelona network.** Barcelona is the second largest city in Spain and capitol of the Catalanian Region. It is the center of a greater metropolitan area that extends over 585 km\(^2\) and has approximately a total of 4 million people or about 68\% of the population in Catalonia. The Barcelona network consist of 1020 nodes, 2522 links and 7922 OD-pairs. The link delays are expressed as highly nonlinear functions of the link flow and have previously demonstrated numerical instability on sequential platforms.
4.5.3.1.2 The Linköping network. This test case is essentially similar to the previous one. However, Linköping, which is Sweden’s fifth largest city is much smaller. In the metropolitan area of Linköping live about 128 000 people. The road network of Linköping consists of 335 nodes, 882 links and 12372 OD-pairs. It is interesting to note that, although the two networks show considerable difference in the number of nodes and links, the number of OD-pairs do not show the same difference. Thus, even for small cities, the number of OD-pairs can exceed by thousand the configuration of any currently available MIMD machine. We, thus, think that the approach to parallelization of the DSD, based on decomposition of the network model by OD-pairs, is promising from a scalability point of view.

4.5.3.1.3 The Winnipeg network. This network of the Canadian city Winnipeg is often used in benchmarking tests (see [56] for a description). It has 1052 nodes, 2836 links and 4344 OD-pairs.

4.5.3.2 Comparison of platforms and networks
Two parallel systems with substantial architecture differences were used: a MIMD computer with distributed memory and 128 processors, four per node, and a cluster formed as a microcomputer consisting of eight computers. The most significant difference (disregarding the number of available processors) between the two parallel machines is the communication performance. The cluster had orders of magnitude better latency and transfer rate which turns beneficial for our algorithm, since the communication requirements are rather large. This is indeed confirmed in §4.5.3.3 where we can observe a better speed-up for the cluster.

A summary of the network data is given in Table 7. The table also specifies the OD-pair to link ratio which is an important factor for the efficiency of our algorithm. In Section 4.5.3.3 we can readily see that the higher ratio the higher efficiency of the parallel algorithm. This implies that the need for high aggregation of OD-pair data is not necessary from the perspective of parallelization, i.e., more detailed networks can be solved without penalty.

4.5.3.3 Numerical experiments
Each experiment was run five times and the solution times reported in the tables below is the average of the five runs. We performed two kinds of tests: i) one snapshot, i.e., finding the user equilibrium (UE) solution for the
network once and ii) 100 snapshots, i.e., finding the network’s UE solution 100 times, where the demand is changing from run to run.

### 4.5.3.3.1 One snapshot

The computational results for one snapshot for the Barcelona, Linköping and Winnipeg network are depicted in Table 8. The speed-ups for the two parallel platforms are given in Figure 10. In Figure 9 we provide a GIS map depicting the flow on the Linköping network according to the one snapshot user equilibrium solution.

Table 8: Wall clock time (not including I/O to disk) in seconds for one snapshot the networks. Requested relative objective error: 0.1 %.

<table>
<thead>
<tr>
<th>Network</th>
<th>Platform</th>
<th>Number of processors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Barcelona</td>
<td>MIMD Computer</td>
<td>44.0</td>
</tr>
<tr>
<td></td>
<td>Multi-computer cluster</td>
<td>30.5</td>
</tr>
<tr>
<td>Linköping</td>
<td>MIMD Computer</td>
<td>49.8</td>
</tr>
<tr>
<td></td>
<td>Multi-computer cluster</td>
<td>36.6</td>
</tr>
<tr>
<td>Winnipeg</td>
<td>MIMD Computer</td>
<td>60.1</td>
</tr>
<tr>
<td></td>
<td>Multi-computer cluster</td>
<td>39.9</td>
</tr>
</tbody>
</table>
4.5.3.3.2 100 snapshots. To simulate the variation in traffic load during a workday (12 hours, with traffic load peaks in the morning, at lunch and in the evening) we implemented a network generator, from which we obtain the OD-pair demand at a given time. The demand is scaled according to Figure 11. The solver is restarted from the previous optimal equilibrium flow for each snapshot.

This test may also be seen as an indicator on the possibility of computing solutions in real-time for large networks as well as in using the code for the solution of more complicated models such as the hierarchical problems mentioned in §3.2.

The computational results for 100 snapshots for the Barcelona, Linköping and Winnipeg network are depicted in Table 9. The speed-ups for the two parallel platforms are given in Figure 12.

4.5.3.3.3 Discussion. It is interesting to notice that although the Linköping network is smaller in size than the other two, better speed-up is obtained for the former. This is due to the higher OD-pair to link ratio of the Linköping network (see Table 7) which clearly affects the result. Indeed, it can be predicted from Table 7 that a higher speed-up should be attainable for
the Barcelona network than for the Winnipeg network. Figure 10 confirms this prediction. The explanation for this behavior is that there is a better computation to communication ratio, since the work load depends in large on the number of OD-pairs and the communication load depends on the number in links.

We can conclude from Table 9, that solutions times between five and ten seconds for each snapshot on all tested networks and platforms are achievable.

From Figure 12 we can observe that the speed-up achieved for the 100 snapshot case is better than for the one snapshot case. This is mainly due to the load balancing which has better effect. Moreover, we see that the speed-up is better for the multi-computer cluster. This is expected since the

Figure 10: Speed-up for one snapshot of all three networks (data from Table 8).
communication characteristics are superior for this platform.

4.6 Other Solution Techniques

Rosen’s gradient projection algorithm [144, 6] has been suggested [147, 148] for solving the routing problem in data networks. They present results for small networks in order to demonstrate the superiority of their approach over the Frank-Wolfe algorithm, however, it is questionable whether gradient projection in the form considered by the authors provides a viable alternative to Frank-Wolfe and Frank-Wolfe-like algorithms for large scale network. Indeed, no particular effort is placed in specializing the algorithm for the underlying network structure, only the Cartesian product is fully utilized.

Cyclic decomposition, which can be interpreted in game theoretic terms as a fictitious play approach to the equilibrium, is particularly suited as a sequential approach to problems of type CPP and consequently to flow assignment problems. The method can be stated as follows for the CPP:

\[ x_i^{k+1} \in \arg \min_{x_i \in \mathcal{X}} f(x_{i-1}^{k+1}, x_i, x_{i+1}^k), \quad \forall i \in \mathcal{N}, \]  

that is, starting with \( x^k \), the next point \( x^{k+1} \) is obtained by cycling through the factors in the Cartesian product. This block coordinate descent approach dates back to Dafermos [44] and Murchland [128]. It does require some specific method to solve the single commodity flow problems in (122).
Table 9: Wall clock time (not including I/O to disk) in seconds for 100 snapshots of the networks. Requested relative objective error for each snapshot: 0.5%.

<table>
<thead>
<tr>
<th>Network</th>
<th>Platform</th>
<th>Number of processors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1  2  4  8  16</td>
</tr>
<tr>
<td>Barcelona</td>
<td>MIMD Computer</td>
<td>2889 1520 822 576 561</td>
</tr>
<tr>
<td></td>
<td>Multi-computer cluster</td>
<td>1970 987 526 356 —</td>
</tr>
<tr>
<td>Linköping</td>
<td>MIMD Computer</td>
<td>7426 4014 2196 1329 1045</td>
</tr>
<tr>
<td></td>
<td>Multi-computer cluster</td>
<td>3898 1954 1035 675 —</td>
</tr>
<tr>
<td>Winnipeg</td>
<td>MIMD Computer</td>
<td>3294 1734 925 708 689</td>
</tr>
<tr>
<td></td>
<td>Multi-computer cluster</td>
<td>2281 1150 633 450 —</td>
</tr>
</tbody>
</table>

Nguyen [130] used the convex simplex method [6, 164], specialized to the underlying network, and reported that the resulting overall approach is about 40% faster than the Frank-Wolfe algorithm. It is demonstrated in [126] that the subproblems in (122) need not be solved to optimality and can be solved rather inaccurately. A few Frank-Wolfe iterations are sufficient. It was suggested in [137] that the cyclic order should be replaced by a so-called Gauss-Southwell order [112] according to which the subproblem in (122) is selected which is furthest from optimality and is then solved. In [137] the subproblem to be selected is determined by solving auxiliary “all-or-nothing assignment” problems, then the selected problem is solved by piecewise linearization [6] of its objective function. It is shown in [126] that if the subproblems (122) are solved approximately using a few Frank-Wolfe iterations, then the subproblem to be solved next is identified without the need to solve auxiliary problems if the properties of the so-called gap function [77], which is immediately available from the Frank-Wolfe subproblems, are appropriately utilized.

As already mentioned, embedding the regularized Frank-Wolfe approach into the disaggregate simplicial decomposition algorithm both for solving the restricted master and for regularizing the Frank-Wolfe subproblems were suggested by [106]. This two-level hybridization results in a most efficient algorithm for the stochastic transportation problem. Indeed, computational results presented in [106] and in [50] confirm the superiority of this hybridization.

77
A plethora of other solution methods, including modifications of the Frank-Wolfe algorithm [4], proximal-point methods [113], projection methods [16, 19, 20], etc., have been proposed. See for instance the survey papers [132, 58, 115] and the book [135].

5 Conclusions

We have reviewed the role of nonlinear optimization in modeling and solving routing and network design problems in data and road networks with fixed topology and we have briefly mentioned new trends in telecommunications.
where nonlinear programming still plays a role.

We have in particular emphasized the role of game theoretic and equilibrium models have played for decades in transportation science and indicated lessons to be learned from that field and to be brought into the field of telecommunications. We expect Wardrop’s equilibrium principle, Nash games and Stackelberg bilevel optimization to play an increasingly important role in telecommunications as globalization, new services and new technology introduce ever increasing competitiveness between firms and more freedom of choice among the users of telecommunication systems.

Further research on routing strategies based on different equilibrium concepts is certain to be a fruitful direction. Elastic demand models provide a suitable instrument in order to motivate or discourage users in their choices of services through appropriate price setting and/or routing mechanisms. We strongly believe that bilevel models based on elastic demand equilibria should be the route to take for setting price and service levels. Network configurations, be it of fixed or ad-hoc topology, must take into consideration the users reaction as the Braess and other paradoxes studied in transportation science have revealed.

Further investigation of game-theoretic modeling for transportation situations where the transport costs are customer costs and where service demanding customers compete for the provided service level at the warehouses, particularly in the case of elastic demand, is may be proven fruitful if combined with competitive location of warehouses. Bilevel decision models with competition in both levels may result from such an approach. Similarly, further investigation of new types of intermediations in production-transportation problems may also be proven fruitful.

In all possible modeling expansions, the requirement of solving convex cost multicommodity flow problem as efficiently as possible, either in its own or in the context of a decomposition or hierarchical approach, still remains. The methods that we have surveyed and analyzed in the present lecture notes constitute a firm basis for understanding the necessary concepts in order to build on further improvements of the existing algorithms or even devise entirely new ones.
Acknowledgments

I thank Dr. Athanasia Karakitsiou for her help in preparing the draft of these lecture notes.

This draft reproduces material from [48, 49, 90, 89, 123, 124, 126].

Unpublished manuscripts and papers in the bibliography are available through CiteSeer, the Autonomous Citation Indexing and Scientific Literature Digital Library, at http://citeseer.ist.psu.edu.

References


[52] J.C. Dunn, “Rate of convergence of conditional gradient algorithms near singular and nonsingular extremals”, *SIAM Journal on Control and Optimization* 17 (1979) 187-211


86


[106] T. Larsson, M. Patriksson, and C. Rydbergren, “Applications of the simplicial decomposition with nonlinear column generation to nonlinear network flows” in *Network Optimization*, D. Hearn et al


91


92


# Contents

1 **Introduction** 2

2 **Brief Overview of the Basic Theory** 5

3 **Models** 14
   3.1 Traffic Assignment and Routing ............................ 14
   3.2 Capacity Assignment and Network Design .................. 21
   3.3 Convex problems in supply chain optimization .............. 24
      3.3.1 The transportation problem with customer competition for the offered service level ................... 25
      3.3.2 The Stochastic Transportation Problem ................ 30
      3.3.3 Production-transportation problem in the presence of an intermediary ..................................... 31
   3.4 Other Applications ........................................... 33

4 **Algorithms** 34
   4.1 The Frank-Wolfe Algorithm .................................. 34
   4.2 Feasible Direction Improvements ............................ 38
   4.3 Simplicial Decomposition and Column Generation ............ 47
   4.4 Hybridization of Simplicial Decomposition and Frank-Wolfe Regularization ................................. 49
      4.4.1 Polynomially bounded dual algorithm for the solution of the subproblems ................................. 54
   4.4.2 Implementation and Computational Results for the Stochastic Transportation Problem ...................... 57
   4.4.3 Tailoring the Algorithm to the Traffic Assignment .... 60
   4.4.4 The hybrid algorithm for the traffic assignment problem ............................ 61
   4.4.5 Computational Results ...................................... 63
   4.5 Distributed System Implementation .......................... 66
      4.5.1 Data structures and distribution ......................... 67
         4.5.1.1 Communication (message passing) .................. 68
         4.5.1.2 Load balancing ................................... 68
         4.5.1.3 Synchronization .................................. 69
      4.5.2 The distributed disaggregate simplicial decomposition algorithm ........................................... 69
         4.5.2.1 Algorithm SPMD-DDSD .............................. 69
4.5.3 Implementation and Computational Results .... 71
  4.5.3.1 Network descriptions .................. 71
    4.5.3.1.1 The Barcelona network .......... 71
    4.5.3.1.2 The Linköping network ......... 72
    4.5.3.1.3 The Winnipeg network .......... 72
  4.5.3.2 Comparison of platforms and networks .. 72
  4.5.3.3 Numerical experiments ............... 72
    4.5.3.3.1 One snapshot .................... 73
    4.5.3.3.2 100 snapshots .................. 74
    4.5.3.3.3 Discussion ...................... 74

4.6 Other Solution Techniques .................. 76

5 Conclusions .................................. 78