UNIVERSITÀ
DEGLI STUDI
FIRENZE

Università degli Studi di Firenze
Dipartimento di Ingegneria dell'Informazione (DINFO)
Corso di Dottorato in Ingegneria dell'Informazione
Curriculum: Automatica, Ottimizzazione e Sistemi Elettrici
SSD MAT/09

Optimization algorithms for network equilibrium problems and transportation planning problems

Candidate
Alessandro Galligari

Supervisors
Prof. Marco Sciandrone

Prof. Fabio Schoen

PhD Coordinator
Prof. Luigi Chisci

ciclo XXX, 2014-2017

Università degli Studi di Firenze, Dipartimento di Ingegneria dell'Informazione (DINFO).

To my family

Acknowledgments

I would like to acknowledge the efforts and input of my supervisors, Prof. Marco Sciandrone and Prof. Fabio Schoen, and all my colleagues and friends of the Global Optimization Laboratory (LabGOL) who were of great help during my research.

My thanks go to my family, who supported me in these three years, in particular in the last one, which has been full of important changes.

Abstract

This thesis is focused on road network problems in the field of transportation planning, which has gained greater attention in the research community in last decades. Modeling the user behavior in order to approximate real traffic scenarios is of interest in many practical applications, such as the planning of new urban networks or the evaluation of enhancements, like widening existing roads or designing new arterial ones.

Furthermore, the estimation of the transportation demand on a multimodal network is of interest since understanding which parts of the network act as generators or attractors of flow have a key role for investments, such as deciding where to build new commercial centers or how to optimize the existing transportation tariff system or building a new one. Transportation planning problems and network equilibrium problems lead to challenging optimization problems that motivate the development of suitable algorithms able to deal with large-scale problems arising in real cases. The main purpose of the work is to develop (in a rigorous way from a mathematical programming point of view) specialized and efficient algorithms, based on a sound convergence theory.

Then, convergent and efficient algorithms have been designed for optimal traffic assignment with both elastic and inelastic demands, formulated as convex, constrained minimization problems. The proposed algorithms can be viewed as block-coordinate descent methods where, at every iteration, a block-component is selected and an inexact minimization is performed. Global convergence results have been established. The extensive experimental and computational study shows that the algorithms are able to obtain very high levels of accuracy on all the large test networks in a limited time (compared with that of state-of-the-art algorithms).

Besides to network equilibrium problems, other planning problems, such as the transportation demand estimation and the design of tariff systems for public transportation networks have been studied. Derivative-free algorithms have been proposed for the estimation of Origin/Destination demands. The extensive set of computational experiments shows the effectiveness of the proposed black-box approach.

Finally, a new local search heuristic has been studied and realized for the planning of zonal tariff systems.

Contents

Chapter 1

Introduction

Traffic assignment consists mainly in the forecast of loading on road arcs in a transportation network. It is a key component of the conventional transport planning process. As it is described in detail by [41], such process can be divided into four steps, also known as the *Four Step Model* (FSM), which derives from *transportation system analysis* as a particular application.

FSM can be examined in a planning paradigm firstly proposed by [40] and expanded later by [17], which is reported in Figure 1.1.

As it is shown in Figure 1.1, the equilibration step is only a part of the whole framework. In such paradigm, the transportation system T, composed by all the transportation elements as well as services, and the activity system A, defined as a set of socio-economic factors (spatial distributions of land use, the demographic and/or economic activity that occurs in those land uses which act as attractors or generators), are exogenous inputs to performance procedures P and demand procedures D, respectively.

The first procedure reflects mode-specific trips, such as person or vehicle, defined at a link level (e.g., freeway vehicle trips per hour) while the second typically produces person trips, defined as the travel from an origin to a destination with the aim of performing some activity, therefore, at a zonal level. Both procedures represents the basic FSM and are interconnected by the path level, since paths comprise sequences of links that connect Origin/Destination (O/D) pairs.

Resolve demand and performance procedures at different spatial levels is the main task of the equilibration process. Starting from the framework in Figure 1.1, the *Four Step Model* (FSM) in Figure 1.2 was developed as a

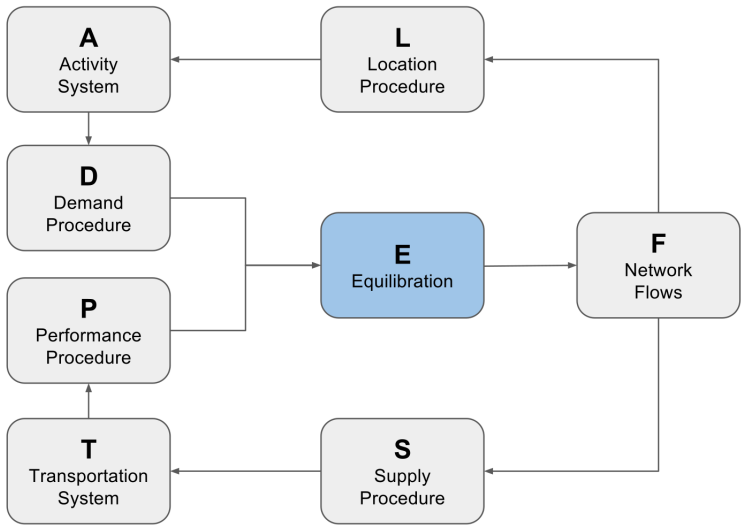


Figure 1.1: The transport planning process framework proposed by [40] and [17].

sequential model to deal with complex realistic applications in which demand functions cannot be estimated directly in order to provide the network flows, together with standard link performance functions and path enumeration.

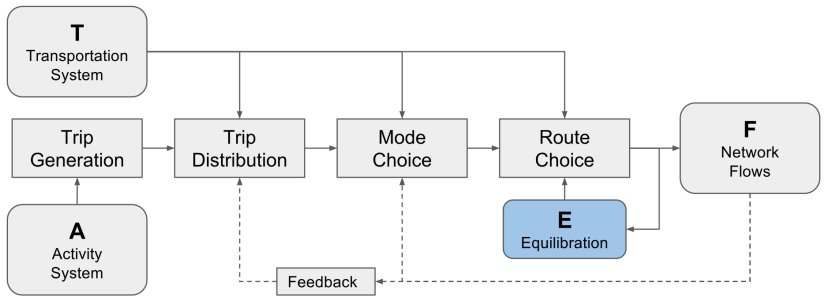


Figure 1.2: The Four Step Model.

The first step is the trip generation, which provides trip frequency based

on an estimation of the propensity to travel. Such trips consist of trip ends, productions and attractions. In trip distribution, trip productions are distributed to match the trip attraction distribution and to reflect the underlying time and/or cost influences, yielding aggregate trip tables of person-trip demands. In mode choice, such tables are then factored in order to reflect relative proportions of trips by alternative modes. Finally, in route choice, each modal trip table is assigned to a mode-specific network based on the properties of the transportation system.

Through the equilibration process, the flow on the arcs of the network can be estimated. In case are used by a feedback in previous steps, the equilibration is called *traffic assignment with elastic demand*, in which trip distribution and mode choice are influenced by the forecasted quantity of load on the network, otherwise is called *traffic assignment with fixed demand*.

Furthermore, the traffic equilibration can be used to evaluate the effect of a variation in the characteristics of the transportation system T in a typical supply procedure S, as it is shown in Figure 1.1. For example, widening an arc road can lead to a modification of the distribution of the traffic in the network. Similarly, the arrangement of flow estimated by the equilibration procedure can be used as an input of a location procedure L, which variates the characteristics of the exogenous contribute of the activity system A. If an area of the network is estimated to be broadly used then it can be more attractive for socio-economic activities.

The equilibration process is the core of the present work and it is mainly studied from an optimization point of view. The main algorithmic contributions have been introduced for the Traffic Assignment Problem (TAP) with fixed demand in Chapter 2, in which a new efficient path-based and globally convergent method has been developed to solve a smooth convex and nonlinear optimization problem with simplex and nonnegativity constraints. Such problem represents the optimization formulation of the *user equilibrium* principle introduced by Wardrop in [62].

In Chapter 3, the same algorithmic solution has been employed in the Traffic Assignment Problem (TAP) with elastic demand, where, as it is illustrated in Figure 1.2, the network flows are used as a feedback for the trip distribution.

Since demand functions can't be derived directly, an estimation of the Origin/Destination (O/D) matrix can be obtained observing the real flows on a set of network links. Assuming that the link flows satisfy the *user*

equilibrium principle, in Chapter 4 the proposed TAP algorithm is used as a black-box in a derivative-free optimization problem with the aim of estimating the O/D demand matrix while reducing the distance between simulated and observed flows.

Another relevant application of the transport planning process is the public transportation planning. In Chapter 5, the zone-based public transportation system firstly proposed in [57] has been described and a new metaheuristic algorithm has been developed to solve the correspondent global NP-hard optimization problem.

Finally, in Chapter 6, the obtained results are discussed.

Chapter 2

New algorithms for the Traffic Assignment Problem

In this chapter, the Traffic Assignment Problem (TAP) is introduced in Section 2.1, while in Section 2.2, the existing contributions about TAP in literature are analyzed. The discussion about the proposed algorithm is organized as follows. In Section 2.3, an introduction to Path Equilibration Algorithms (PEAs) is reported. The proposed algorithms ISMO and ISMO-ACG are presented in Section 2.4 and 2.5 respectively, while in Section 2.6 two non-monotone versions of ISMO-ACG are considered. In Section 2.7, a general path-based framework is introduced where several equilibration strategies can be employed in the same column generation framework. Finally, in Section 2.8, an extensive analysis of the obtained results is reported.

2.1 Preliminary background

The aim of a network equilibrium model is to predict the link flows of a network which depend on the paths connecting Origin/Destination pairs (from now on, O/D pairs) chosen by the users of the network. This challenging problem has a great importance since finding fast algorithms to solve it is of interest in different practical applications and has received attention in the last 50 years motivating the development of many algorithms to tackle the difficulty due to the size of real problems.

The traffic assignment problem mainly relies on the *user equilibrium* definition stated by Wardrop in [62].

Definition 1 (User Equilibrium)**.** For each O/D pair, the costs of all paths actually used, i.e., with nonzero flow, are equal or less than those which would be experienced by a single vehicle on any unused path.

The definition stated in Definition 1 models a non-cooperative behavior of users on the network, that is, each user on the network chooses independently its route on the base of its perception about which could be the fastest path from its origin to its destination.

Before modeling it as an optimization problem, let's introduce the notation that will be used throughout the chapter.

Let $G = (N, A)$ be the network graph with N as the set of nodes and A as the set of arcs. Each node $u \in N$ can be a crossroad, a place or even an abstract centroid which acts as a generator or an attractor of flow. In the same way, an arc $a \in A$ can be an existing road or a virtual link between centroids and nodes in the network.

Let $P \subseteq N \times N$ be the set of all O/D pairs for which is defined a travel demand $D_p > 0$. Generally, the O/D matrix $D = \{D_p\}$, $p = (u, v)$, can be viewed as a sparse matrix since the transportation demand is often defined for O/D pairs connecting centroids, while network nodes such crossroads are rarely sources or destinations of flow.

With k we indicate a path, i.e., a sequence u_i, $u_i \in N$, of nodes. The finite set of paths connecting an O/D pair $p \in P$ is denoted with K_p, in which only a-cyclic (simple) paths are considered.

The set of all paths is indicated with K,

$$K = \bigcup_{p \in P} K_p \tag{2.1}$$

For convenience, let $n_p = |K_p|$ and $n = |K|$. We denote by $x \in \mathbb{R}^n$ the vector of path flows and with x_k the flow on path $k \in K$. For each O/D pair $p \in P$ we define its block of path flows variables with $x_{(p)} \in \mathbb{R}^{n_p}$ and its i-th element as $x_{(p),i}$.

Through the definition of an arc-path incidence matrix such as

$$\delta_{ak} = \begin{cases} 1 & \text{if arc } a \in A \text{ belongs to path } k \in K \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

for all $a \in A$ and $k \in K$, the arc flow v_a is a function of the vector of path flows x and it's obtained as follows

$$v_a(x) = \sum_{k \in K} \delta_{ak} x_k, \qquad \forall\ a \in A. \tag{2.3}$$

Each arc $a \in A$ of the network has associated a cost function $s_a(\cdot)$ which, in general, is dependent on the overall flow distribution in the network. In this thesis, however, $s_a(\cdot)$ is a function only of the flow $v_a(x)$ of the arc $a \in A$, thus costs are separable. Let's summarize in the following assumption.

Assumption 2.1 (Separable costs)**.** *The arc costs $s_a(\cdot)$ are separable, $\forall\ a \in A$, i.e., $s_a(\cdot)$ depends only on the flow $v_a(x)$ on arc $a \in A$.*

Moreover, we assume additivity of path cost functions denoted by $s_k(\cdot)$.

Assumption 2.2 (Additivity)**.** *The cost $s_k(\cdot)$ of a path $k \in K$ is the sum of the cost of each arc belonging to the path, i.e.,*

$$s_k(x) = \sum_{a \in A} \delta_{ak} s_a(v_a(x)). \tag{2.4}$$

Finally, we state the following assumption.

Assumption 2.3 (Strictly increasing cost)**.** *The arc cost $s_a(\cdot)$, $\forall\ a \in A$, is a continuously differentiable and strictly increasing function.*

The above assumptions allow formulating the TAP as a linearly constrained convex programming problem. In particular, there are two formulations of the problem, the first one is the so-called *arc-node* formulation, while the second one is the *arc-route* formulation.

The former representation has a polynomial number of arc flow variables and flow conservation constraints, whereas the latter has a simpler constraint structure, but an exponential number of path flow variables, which must be enumerated iteratively. In this thesis, we will refer to the arc-route formulation, which is presented in the following proposition.

Proposition 2.4 (TAP convex programming problem)**.** *With the assumptions stated above, the TAP arc-route formulation is given as follows*

$$\begin{aligned} \min_{x} \quad & f(x) = \sum_{a \in A} \int_0^{v_a(x)} s_a(t)\, dt & \\ s.t. \quad & \sum_{i=1}^{n_p} x_{(p),i} = D_p, & \forall\ p \in P \\ & x_{(p)} \geq 0, & \forall\ p \in P \end{aligned} \tag{2.5}$$

It's easy to show that the partial derivative $\nabla_k f(x)$ of the objective function in (2.5) with respect to the path flow variable x_k is the cost $s_k(x)$ of path $k \in K$.

Finally, the problem (2.5) can be rewritten in a more compact way as follows

$$\begin{aligned} \min_x \quad & f(x) \\ s.t. \quad & x \in \mathcal{F} = \prod_{p \in P} \mathcal{F}_p \end{aligned} \tag{2.6}$$

where $f(\cdot)$ is a convex continuously differentiable function and $\mathcal{F}$ is the cartesian product of simplices $\mathcal{F}_p$ defined as follows

$$\mathcal{F}_p = \{x_{(p)} \in \mathbb{R}^{n_p} : \sum_{i=1}^{n_p} x_{(p),i} = D_p,\ x_{(p)} \geq 0\}. \tag{2.7}$$

2.2 Related work

The algorithms proposed in the wide literature can be divided into three main classes (for a suitable treatment and for references to the other methods one may refer to [52]).

Link-based. The algorithms of this class, where the optimization variables are link flows, are mainly the Frank-Wolfe method (see [19]) and its variants. It is one of the most common algorithm for the TAP, since its memory requirement is relatively small. However, the main drawback of the method lies in its slow convergence rate. Several modifications (see [20, 34, 43, 53]) have been proposed to accelerate the speed of convergence trying to avoid the zig-zagging which typically occurs in later iterations (see [50]) of the method.

Bush-based. In bush-based methods the optimization variables are link flows associated with the different origins; see, e.g., [6,7,13,25,44,45,64].

Path-based. This class of methods has gained a greater attention in recent years thanks to the development of efficient algorithms which address properly the problem of optimizing with respect to path variables, which are hard to enumerate a priori.

Indeed, the common feature of path-based algorithms is the adoption of a *column generation* approach in order to avoid storing all possible

paths for each O/D pair. By the column generation strategy, only a subset of *active* path variables are kept in memory, those whose current flow value is positive, and new paths are iteratively generated, when needed, to reach optimality. The column generation procedure involves the computation of the shortest route, which often constitutes the most computationally intensive part of a path-based algorithm.

When performing column generation, the problem (2.6), at each iteration t, becomes the following

$$\begin{aligned} \min_{x} \quad & f(x) \\ s.t. \quad & x \in \mathcal{F}^t = \prod_{p \in P} \mathcal{F}_p^t \end{aligned} \tag{2.8}$$

where $\mathcal{F}_p^t$ is a lower-dimensional subset (restriction) of the factor set $\mathcal{F}_p$.

These methods mainly differ in:

(i) whether or not to adopt a column generation strategy,

(ii) whether or not to adopt a decomposition strategy;

(iii) whether or not to exactly solve the problem (2.8) (or the corresponding subproblems in a decomposition framework);

(iv) the employed minimization method.

Many path-based algorithms have been proposed in the literature, see, e.g., [11, 16, 18, 32, 35].

In this work, we refer mainly to path-based algorithms. Many of these can be embedded in a general framework where, at each iteration, a feasible and descent direction is computed, and the current solution is updated by performing a step along this direction.

The so-called *Path Equilibration Algorithms* (PEAs) are path-based methods where the search direction contains only two nonzero elements, those corresponding to two paths of the considered O/D pair with different costs. The stepsize along this direction is computed with the aim of equalizing the two costs.

PEAs belong to the class of Gauss-Seidel decomposition methods and have a long history. The first contribution can be found in [11], in which,

at each iteration, a quadratic approximation of the objective function along the search direction is computed.

In [18], the same decomposition strategy is performed and the minimum and maximum cost routes are optimized in a single inner iteration for each O/D pair. In a later work (see [16]), the inner optimization is performed with an adaptation of the Rosen's projected gradient algorithm.

The proposed algorithms are both PEAs and will be presented after a preliminary introduction about the PEAs framework.

2.3 Path Equilibration Algorithms (PEAs)

Path Equilibration Algorithms (PEAs) can be viewed as cyclic block-coordinate (or Gauss-Seidel) methods. At every iteration, a block-component (corresponding to an O/D pair $p \in P$) is selected in cyclic order and a suitable quantity of flow is shifted from a (maximum) cost used route to a lower (minimum in some cases) cost route.

In order to formally define a general framework of PEAs, given an O/D pair $p \in P$, we introduce the definition of a direction $d^{i,j}_{(p)} \in \mathbb{R}^n$ with only two nonzero components in correspondence to the indices $i, j \in \{1, \ldots, n_p\}$, $i \neq j$,

$$d^{i,j}_{(p),h} = \begin{cases} 1 & \text{if } h = i \\ -1 & \text{if } h = j \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

It can be easily shown that $d^{i,j}_{(p)}$ is a feasible direction at a point $\bar{x} \in \mathcal{F}$ if and only if $\bar{x}_{(p),j} > 0$. Moreover, since f is convex, $d^{i,j}_{(p)}$ is a descent direction if and only if

$$\nabla_i f(\bar{x}) < \nabla_j f(\bar{x}).$$

Indeed, we have

$$\begin{aligned} &\nabla f(\bar{x})^\top d^{i,j}_{(p)} < 0 \Leftrightarrow \\ &\nabla_i f(\bar{x}) - \nabla_j f(\bar{x}) < 0 \Leftrightarrow \\ &\nabla_i f(\bar{x}) < \nabla_j f(\bar{x}) \end{aligned} \tag{2.10}$$

A sketch of a PEA is reported below in Algorithm 1. PEAs differ in:

(a) the choice of the pair $(i^\star, j^\star)$;

(b) the computation of the stepsize α.

Algorithm 1: Path equilibration framework

Data: a feasible point $x^0 \in \mathcal{F}$
$t \leftarrow 0$;
while x^t *is not a solution* **do**
  for $p \in P$ **do**
    select a pair of indices $(i^\star, j^\star)$ such that $\nabla_{i^\star} f(x^t) < \nabla_{j^\star} f(x^t)$;
    $d^t \leftarrow d_{(p)}^{i^\star, j^\star}$;
    compute the stepsize α^t along $d_{(p)}^{i^\star, j^\star}$;
    $x^{t+1} \leftarrow x^t + \alpha^t d_{(p)}^{i^\star, j^\star}$;
    $t \leftarrow t + 1$;
  end
end

Concerning point (a), we observe that the choice of the pair $(i^\star, j^\star)$ must guarantee that the corresponding search direction $d_{(p)}^{i^\star, j^\star}$ is a feasible and descent direction at x. Furthermore, as already explained, it is not reasonable in practice to store all the possible variables $x_{(p),h}$, with $h = 1, \ldots, n_p$. Therefore, a column generation approach is usually adopted and is based on the idea of adding new paths when necessary. In particular, given an O/D pair $p \in P$ and a solution x^t at iteration t, let us introduce the index set

$$I_p = \{1, \ldots, n_p\}$$

and

$$I_p^+(x^t) = \{i \in I_p : x_{(p),i}^t > 0\},$$

i.e., the set of active path indices at iteration t. Thus, the algorithm requires to store only the variables that have positive flow value. Then, the pair selection rule usually employed in PEAs is the following

$$\begin{aligned} i^\star &\in \arg \min_{i \in I_p} \nabla_i f(x^t) \\ j^\star &\in \arg \max_{j \in I_p^+(x^t)} \nabla_j f(x^t) \end{aligned} \tag{2.11}$$

Recalling the meaning of partial derivatives, this is equivalent to selecting the minimum and the maximum cost routes, where the latter has to be active since $d_{(p)}^{i^\star, j^\star}$ is a feasible direction.

Paths corresponding to indices $i^\star$ and $j^\star$ are then equalized and $i^\star$ will be added to $I_p^+(x^t)$ if necessary, while $j^\star$ may leave the working set $I_p^+(x^t)$ if the flow has gone to zero.

As regards point (b) concerning the computation of the stepsize α along the direction $d_{(p)}^{i^\star,j^\star}$, the following rules have been proposed:

(i) (see [11]) a quadratic model of the objective function is used, i.e.,

$$f(x+\alpha\, d_{(p)}^{i^\star,j^\star}) \approx m_q(\alpha)$$
$$m_q(\alpha) = f(x) + \alpha \nabla f(x)^\top d_{(p)}^{i^\star,j^\star} + \frac{1}{2}\alpha^2 d_{(p)}^{i^\star,j^\star\,\top} \nabla^2 f(x)\; d_{(p)}^{i^\star,j^\star}, \tag{2.12}$$

and α is set as follows

$$\alpha \in \arg \min_{0\le\alpha\le x_{(p),j^\star}} m_q(\alpha) = min\{x_{(p),j}, \alpha^\star\}, \tag{2.13}$$

where $\alpha^\star$ is the unconstrained minimum of $m_q(\alpha)$, i.e.,

$$\alpha^\star = -\frac{\nabla f(x)^\top d_{(p)}^{i^\star,j^\star}}{d_{(p)}^{i^\star,j^\star\,\top} \nabla^2 f(x)\; d_{(p)}^{i^\star,j^\star}} \tag{2.14}$$

Thus, $\alpha^\star$ is computed using second-order information. Recalling the form of the search direction $d_{(p)}^{i^\star,j^\star}$, $\alpha^\star$ is explicitly given as follows

$$\alpha^\star = -\frac{\nabla_{i^\star} f(x) - \nabla_{j^\star} f(x)}{\nabla^2_{i^\star} f(x) + \nabla^2_{j^\star} f(x) - 2\nabla^2_{i^\star j^\star} f(x)} \tag{2.15}$$

and from Problem (2.5), with a few steps, we have

$$\alpha^\star = -\frac{\displaystyle\sum_{a\in i^\star\setminus(i^\star\cap j^\star)} s_a(v_a(x)) - \sum_{a\in j^\star\setminus(i^\star\cap j^\star)} s_a(v_a(x))}{\displaystyle\sum_{a\in i^\star\setminus(i^\star\cap j^\star)} \frac{\partial s_a}{\partial a}(v_a(x)) + \sum_{a\in j^\star\setminus(i^\star\cap j^\star)} \frac{\partial s_a}{\partial a}(v_a(x))} \tag{2.16}$$

It can be seen from (2.16) that only non-common arcs of paths $i^\star$ and $j^\star$ are used for the computation of $\alpha^\star$;

(ii) (see [18]) the stepsize α is computed by solving a one-dimensional problem, i.e.,

$$\alpha \in \arg \min_{0\le\alpha\le x_{(p),j^\star}} f(x+\alpha d_{(p)}^{i^\star,j^\star});$$

(iii) (see [52]) the stepsize is computed by the Armijo backtracking line search based on the acceptance condition of the form

$$f(x + \alpha d_{(p)}^{i^\star,j^\star}) \leq f(x) + \gamma\alpha\nabla f(x)^\top d_{(p)}^{i^\star,j^\star},$$

with $\gamma \in (0,1)$.

2.4 ISMO

In [12] a convergent and efficient PEA was developed, which introduces a novel strategy for the column generation procedure. The main idea is to avoid, for each O/D pair, the generation of a new path at each iteration t. Since it involves the computation of the shortest route (see the first row in (2.11)), the column generation is an expensive computational task and it represents most of the time spent during the optimization. The resulting pair of selected indices $(i^\star, j^\star)$ is then relative to active paths belonging to the current working set $I_p^+(x^t)$ and the resulting direction $d_{(p)}^{i^\star,j^\star}$ is such that the equilibration is performed among between existing used paths.

Thus, (2.11) is modified as follows

$$\begin{aligned} i^\star &\in \begin{cases} \arg\min\limits_{i \in I_p} \nabla_i f(x^t) & \text{if column generation is applied} \\ \arg\min\limits_{i \in I_p^+(x^t)} \nabla_i f(x^t) & \text{otherwise} \end{cases} \\ j^\star &\in \arg\max_{j \in I_p^+(x^t)} \nabla_j f(x^t) \end{aligned} \tag{2.17}$$

However, in order to guarantee the convergence (discussed in Section 2.4.1), the procedure must be applied within a prefixed finite number L of iterations where the given block-component $x_{(p)}$ is selected.

For what concerns the computation of the stepsize α, a Quadratic Line Search (QLS) (described in Appendix A) has been used which guarantees that the distance between successive points tends to zero, which is a usual requirement in the context of decomposition methods to guarantee global convergence properties.

None of the reported rules in the previous Section, indeed, ensure the convergence without further assumptions. As an example, the standard Armijo

line search does not satisfy the above requirement, while in [11] the convergence has been proved by assuming that the arc cost functions are sufficiently close to some strictly convex quadratic function.

The lack of a proper convergence analysis in most of the algorithms in literature has been pointed out in [51], since most researchers in the field come from an area different from mathematical programming. As a consequence, very often important mathematical programming issues related to the convergence theory are not suitably investigated and analyzed. For instance, in the seminal paper [13] that led to the current state-of-the-art algorithms, the author clearly wrote: "For the Reader requiring more rigorous support than this writer can provide for mathematical claims made here, we suggest Ahuja et al. (1993), Bertsekas (1995), Patriksson (1994) and Rockafellar (1984)."

The main purpose of the work presented in [12] is to develop (in a rigorous way from a mathematical programming point of view) a specialized and efficient algorithm, based on a sound convergence theory, for network equilibrium problems with arc cost functions that are separable.

Summarizing, the peculiarities of the proposed algorithm, as a PEA, are the following:

- it performs a Gauss-Seidel decomposition;
- it sequentially and inexactly solves subproblems of two variables, which is the minimal number of variables that can be updated, by performing suitable line searches along feasible and descent directions;
- for each block-component $x_{(p)}$, with the aim of reducing the computational effort, the column generation procedure is not applied for each block-component at every iteration but within a prefixed number L of iterations.

The algorithm, called Inexact Sequential Minimal Optimization (ISMO), is formally described below in Algorithm 2.

In line 13, an initial stepsize α_0^t is computed using first-order information, which, in general, may be unfeasible. However, in Algorithm QLS reported in Appendix A, the initial stepsize of the procedure is computed using the minimum value between α_0^t and β^t in order to mantain the feasibility of the update.

According to the instructions of steps 17–18, whenever the computation of the shortest route for the $p-$th O/D pair is performed and the step length

Algorithm 2: Inexact Sequential Minimal Optimization (ISMO) Algorithm

Data: A feasible point $x^0 \in \mathcal{F}$, an integer $L > 0$, $c > 0$

$t \leftarrow 0$;
$l_p \leftarrow 1$, $I_p^+(x^t) \leftarrow \{i \in I_p : x^t_{(p),i} > 0\}$, $\forall\ p \in P$;
while x^t *is not a solution* **do**
  for $p \in P$ **do**
    if $l_p = 1$ **then**
      $i^\star \leftarrow \arg\min_{i \in I_p} \nabla_i f(x^t)$;
    else
      $i^\star \leftarrow \arg \min_{i \in I_p^+(x^t)} \nabla_i f(x^t)$;
    end
    $j^\star \leftarrow \arg \max_{i \in I_p^+(x^t)} \nabla_i f(x^t)$;
    $d^t \leftarrow d^{i^\star, j^\star}_{(p)}$;
    $\beta^t \leftarrow x^t_{(p),j^\star}$;
    $\alpha_0^t \leftarrow -c \nabla f(x^t)^\top d^{i^\star, j^\star}_{(p)}$;
    $\alpha^t \leftarrow QLS(x^t, d^t, \alpha_0^t, \beta^t)$;
    $x^{t+1} \leftarrow x^t + \alpha^t d^t$;
    $I_p^+(x^{t+1}) \leftarrow \{i \in I_p : x^{t+1}_{(p),i} > 0\}$, $I_q^+(x^{t+1}) = I_q^+(x^t)$, $q \neq p$;
    if $\alpha^t = \beta^t$ *and* $l_p = 1$ **then**
      $l_p = 1$;
    else
      $l_p \leftarrow \mathrm{mod}\ (l_p, L) + 1$;
    end
    $t \leftarrow t + 1$;
  end
end

α^t is equal to the maximum feasible step length β^t, at the next iteration where the same O/D pair is considered, the column generation procedure must be applied again. This plays a theoretical role for ensuring global convergence properties and is exploited in the proof of Proposition 2.8 discussed in the next section.

2.4.1 Convergence analysis

Let us consider problem (2.6). Given a feasible point $\bar{x}$, we denote by $D(\bar{x})$ the set of feasible directions at $\bar{x}$. A feasible point $\bar{x}$ is a solution of problem (2.6) if and only if

$$\nabla f(\bar{x})^\top d \geq 0 \qquad \forall\, d \in D(\bar{x}). \tag{2.18}$$

The following result can be easily proved.

Proposition 2.5. *A feasible point x^* is a solution of Problem* (2.6) *if and only if for all $p \in P$ we have*

$$\nabla f(x^*)^\top d^{i,j}_{(p)} \geq 0 \qquad \forall\, d^{i,j}_{(p)} \in D(x^\star). \tag{2.19}$$

The next result was stated in [36] (assertion (i) of Proposition 4).

Proposition 2.6. *Let $\{x^k\}$ be a sequence of feasible points such that $x^k \to x^\star$ for $k \to \infty$. Then, for k sufficiently large we have*

$$D(x^\star) \subseteq D(x^k). \tag{2.20}$$

Before we can prove the global convergence of ISMO, we must state the following technical result.

Proposition 2.7. *For each $p \in P$, let T_p be the subset of iterates where the index p is selected. Then, there exists a number M_p such that, for every $t \in T_p$, there exists an index $m(t)$ such that $t + m(t) \in T_p$,*

$$0 \leq m(t) \leq M_p \tag{2.21}$$

and

$$\alpha^{t+m(t)} < \beta^{t+m(t)}, \tag{2.22}$$

where $\beta^{t+m(t)}$ is the maximum feasible step length along d^t, that is

$$\beta^{t+m(t)} = x^{t+m(t)}_{(p),j^{t+m(t)}}. \tag{2.23}$$

Proof. Given $p \in P$, we have

$$T_p = \{p-1, p-1+|P|, p-1+2|P| \dots, \}. \tag{2.24}$$

The instructions of the algorithm imply

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

For simplicity's sake, and without loss of generality, we can assume

$$f(x^{t+1}) < f(x^t). \tag{2.25}$$

For each $k \in T_p$ let $\|x^t_{(p)}\|_0$ be the number of non–zero components of $x^t_{(p)}$. We will assume that, at every iteration $h \geq t$, with $h \in T_p$, the algorithm will only perform iterations where $\alpha^h = \beta^h$, and show that this cannot happen for more than a prefixed number M_p of iterations. After every iteration h, we have

$$x^h_{(p),j^h} > 0 \qquad x^{h+1}_{(p),j^h} = 0.$$

Two cases can happen

1. $x^h_{(p),i^h} = 0$. Then $\|x^{h+1}_{(p)}\|_0 = \|x^h_{(p)}\|_0$. Furthermore, $x^{h+1}_{(p),i^h} = x^h_{(p),j^h}$ and $x^{h+1}_{(p),j^h} = x^h_{(p),i^h}$, so we can see that the vector $x^{h+1}_{(p)}$ is a permutation of $x^h_{(p)}$.

2. $x^h_{(p),i^h} > 0$. Then $\|x^{h+1}_{(p)}\|_0 = \|x^h_{(p)}\|_0 - 1$.

As $x^h_{(p)} \in \mathbb{R}^{n_p}$ and there are $n_p!$ permutations for a vector of size n_p, from (2.25) the first case cannot happen for more than $n_p!$ consecutive iterations where the index p is selected. Moreover, $\|x^h_{(p)}\|_0$ cannot be lower than zero, and can never increase, so the second case can happen at most for a number of iterations (where the index p is selected) equal to n_p. Then, recalling (2.24), we have $\alpha^h < \beta^h$ for some $h \in T_p$ and $h \leq t + |P| \cdot n_p \cdot (n_p!)$, and hence we obtain

$$M_p = |P| \cdot n_p \cdot (n_p!). \tag{2.26}$$

□

The global convergence of the ISMO Algorithm is stated in Proposition 2.8.

Proposition 2.8. *Let $\{x^t\}$ be the sequence generated by Algorithm ISMO. Then $\{x^t\}$ admits limit points and each limit point is a solution of Problem (2.6).*

Proof. The instructions of the algorithm imply that the sequence of points $\{x^t\}$ belongs to the feasible set, which is a compact set, so that $\{x^t\}$ admits limit points. Furthermore, we have

$$f(x^{t+1}) = f(x^t + \alpha^t d^t) \leq f(x^t), \tag{2.27}$$

so that, we can recall condition (A.3) of Algorithm QLS, and we can write

$$\lim_{t\to\infty} \|x^{t+1} - x^t\| = 0. \tag{2.28}$$

From (2.28), as $\|d^t\| = \sqrt{2}$, we get

$$\lim_{t\to\infty} \alpha^t = 0. \tag{2.29}$$

Let $x^\star$ be a limit point of $\{x^t\}$, i.e., there exists an infinite subset $T \subseteq \mathbb{N}$ such that

$$\lim_{t\in T, t\to\infty} x^t = x^\star. \tag{2.30}$$

By contradiction, let us assume that $x^\star$ is not a solution. From Proposition 2.5, there exist $\hat{p} \in P$, $\hat{i}, \hat{j} \in I_{\hat{p}}$ such that

$$\nabla f(x^\star)^\top d_{(\hat{p})}^{\hat{i},\hat{j}} < 0. \tag{2.31}$$

Taking into account the instructions of the algorithm it follows that, for all $t \in T$, there exists an integer $l(t) \leq |P| \cdot L$ such that

$$p^{t+l(t)} = \hat{p},$$

and

$$i^{t+l(t)} \in \arg\min_{i\in I_{\hat{p}}} \nabla_i f(x^{t+l(t)}). \tag{2.32}$$

Let

$$T = \{t_1, t_2, \ldots\}$$

and define

$$T_1 = \{t_1 + l(t_1), t_2 + l(t_2), \ldots\}.$$

Note that, by definition, $T_1 \subseteq T_{\hat{p}}$, being $T_{\hat{p}}$ the subset of iterates where the index $\hat{p}$ is selected.

As it holds $l(t_h) \leq L \cdot |P|$ for $h = 1, 2, \ldots, \infty$, recalling (2.28), we have

$$\lim_{t \in T_1, t \to \infty} x^t = x^\star. \tag{2.33}$$

For each $t \in T_1$, let $m(t)$ be the first nonnegative integer such that $t + m(t) \in T_{\hat{p}}$ and

$$\alpha^{t+m(t)} < \beta^{t+m(t)}. \tag{2.34}$$

Proposition 2.7 ensures the existence of such an index. Further we have $m(t) \leq M_{\hat{p}}$, hence we can write

$$\lim_{t \in T_1, t \to \infty} x^{t+m(t)} = x^\star. \tag{2.35}$$

The definition of the index $m(t)$ implies that either $m(t) = 0$ and

$$\alpha^t < \beta^t,$$

or $m(t) \geq 1$ and

$$\alpha^{t+j} = \beta^{t+j}$$

for $t = 0, \ldots, m(t) - 1$. In the former case, as $t \in T_1$, from (2.32) we obtain

$$i^t \in \arg\min_{i \in I_{\hat{p}}} \nabla_i f(x^t).$$

In the latter case, recalling again that $t \in T_1$, taking into account the test and the instruction at steps 17–18, we have

$$i^{t+j+1} \in \arg\min_{i \in I_{\hat{p}}} \nabla_i f(x^{t+j+1}), \tag{2.36}$$

for $j = 0, \ldots, m(t) - 1$. Then, in all cases we can write

$$i^{t+m(t)} \in \arg\min_{i \in I_{\hat{p}}} \nabla_i f(x^{t+m(t)}). \tag{2.37}$$

Therefore, for all $t \in T_1$ we have $t + m(t) \in T_1$ and

$$\nabla f(x^{t+m(t)})^\top d^{t+m(t)} \leq \nabla f(x^{t+m(t)})^\top d^{i,j}_{(\hat{p})} \qquad \forall\, d^{i,j}_{(\hat{p})} \in D(x^{t+m(t)}). \tag{2.38}$$

Since $d^{t+m(t)}$ belongs to a finite set, we can extract a further subset of T_1, that we relabel again with T_1, such that $d^{t+m(t)} = d^\star$ for all $t \in T_1$. Furthermore, (2.38) and Proposition 2.6 imply

$$\nabla f(x^{t+m(t)})^\top d^\star \leq \nabla f(x^{t+m(t)})^\top d^{\hat{i},\hat{j}}_{(\hat{p})} < 0. \tag{2.39}$$

From (2.29) and (2.34) we get

$$\alpha^{k+m(k)} < \min\{\lambda, \beta^{k+m(k)}\},$$

where λ is the fixed initial step length in Algorithm QLS. Therefore, the instructions of Algorithm QLS imply

$$f\left(x^{t+m(t)} + \frac{\alpha^{t+m(t)}}{\delta} d^\star\right) > f(x^{t+m(t)}) - \gamma \left(\frac{\alpha^{t+m(t)}}{\delta}\right)^2 \|d^\star\|^2. \tag{2.40}$$

From (2.40), using the Mean Value Theorem, we can write

$$\nabla f(\xi^{t+m(t)})^\top d^\star \geq -\frac{\alpha^{t+m(t)}}{\delta} \|d^\star\|^2,$$

where

$$\xi^{t+m(t)} = x^{t+m(t)} + \omega^t \frac{\alpha^{t+m(t)}}{\delta} d^\star$$

and $\omega^t \in (0,1)$. Then, taking the limits for $t \in T_1$ and $t \to \infty$, recalling (2.35), (2.39) and (2.29), we obtain

$$\lim_{t \in T_1, t \to \infty} \nabla f(x^{t+m(t)})^\top d^\star = \nabla f(x^\star)^\top d^\star = 0. \tag{2.41}$$

Therefore, using (2.41) and (2.39), we obtain

$$\nabla f(x^\star)^\top d^{\hat{i},\hat{j}}_{(\hat{p})} \geq 0,$$

which contradicts (2.31).

□

2.5 ISMO-ACG

As it is shown in Section 2.8, the column generation strategy adopted by ISMO is able to obtain valuable performances compared to the case where

shortest paths are recomputed at each iteration. Moreover, global convergence is proven thanks to the adoption of the Quadratic Line Search procedure. However, the algorithm shows difficulties in achieving high precisions when the size of the network increases. It turns out that when the algorithm reaches a given precision of the solution, the initial stepsize α_0^t computed with first-order information becomes unsuitable from a numerical point of view. The line search procedure, indeed, is then not able to reduce properly the stepsize in order to ensure an update which actually decreases the objective function.

This is due to a numerical issue which occurs when dealing with finite precision floating point representations. It can be shown that, when the initial stepsize becomes smaller as the precision of the solution increases, the line search procedure tends to accept it without requiring any reducing iteration. In other words, the line search becomes useless when reaching the limits of the floating representation. Thus, the solution is updated using only first-order information and this leads to an oscillatory behavior.

Such a behavior is confirmed in [12] when a numerical workaround is adopted which consists in reducing the stepsize α^t by a given factor $\tau \in (0, 1)$ when the oscillation event occurs. The adoption of this trick leads, indeed, to a significant increase of the solution precision in closest iterations. However, as it is expected, it does not resolve the issue at all.

The development of the algorithm described in [22] concerns mainly two different aspects:

(i) solving the numerical issues experienced by ISMO while computing the stepsize α^t along d^t with the aim of achieving high solutions precision;

(ii) developing a new column generation strategy which adaptively compute shortest paths when needed, in order to increase the efficiency of method.

For what concern (i), the rule employed in [11] and described in Section 2.3 is suitably combined with QLS, that is, the optimum of the quadratic approximation of the objective function along the search direction d^t is adopted as the initial line step.

Regarding point (ii), let's recall again the idea underlying the approach proposed in [12] and described in Section 2.4. The column generation strategy, which adds variables when necessary, involves the computation of the

shortest route, which constitutes the most computationally intensive part of path-based algorithms. For each block-component p, with the aim of reducing the computational effort, in the algorithm ISMO the column generation procedure is not applied at every iteration; in order to guarantee the convergence, the procedure must be applied within a prefixed number L of iterations where the given block-component is selected.

We observe that, in ISMO, the parameter L is the same for all the O/D pairs and is fixed across iterations. The strategy designed in [22] associates different parameters to different O/D pairs. Thus, the shortest path between origin and destination for $p \in P$ is computed once every L_p iterations and this value may be updated during the optimization.

When a new path is added at iteration t for the O/D pair p, it is expected that, moving a suitable quantity of flow from the most costly path to the new path, which is the shortest path, a larger decrement in the quadratic approximation (needed for the computation of the step as in [11]) is obtained compared to the one achieved at the previous iteration, when column generation has not been performed. The benefit deriving from the adoption of the column generation is evaluated by measuring the reduction of the quadratic model, i.e., by computing

$$\Delta_p^{t+1} = \alpha^t \nabla f(x^t)^\top d_{(p)}^{i^\star,j^\star} + \frac{1}{2}(\alpha^t)^2 d_{(p)}^{i^\star,j^\star\,\top} \nabla^2 f(x^t)\; d_{(p)}^{i^\star,j^\star} \tag{2.42}$$

where the stepsize α^t is given by formula (2.14). If Δ_p^{t+1} is greater than Δ_p^t, then the column generation is considered effective, and hence the value of parameter L_p is decreased (i.e., column generation for O/D pair p will be performed more frequently). Otherwise, the value of L_p is increased since the benefit of the column generation is not considered valuable. In any case L_p should be bounded in $[L_{min}, L_{max}]$, $0 < L_{min} < L_{max} < \infty$, in order to maintain the convergence properties of the algorithm, which derive from ISMO and are briefly discussed below.

The algorithm, which has been called Inexact Sequential Minimal Optimization with Adaptive Column Generation (ISMO-ACG), is summarized in Algorithm 3. In order to facilitate the readability, some steps needed for the convergence analysis reported for ISMO have been removed and the notation simplified.

Note that, for each O/D pair p, the shortest path is computed every L_p iterations (see lines 5–9), furthermore, the value of L_p can be increased or

Algorithm 3: Inexact Sequential Minimal Optimization with Adaptive Column Generation (ISMO-ACG)

Data: $x^0 \in \mathcal{F}$, $L_{min}, L_{max} \in \mathbb{N} : 0 < L_{min} < L_{max} < \infty$

$t \leftarrow 0$;
$L_p \leftarrow L_{min}$, $t_p \leftarrow t$, $\forall\ p \in P$;
while x^t *is not a solution* **do**
  for $p \in P$ **do**
    if $(t - t_p)$ ***mod*** $L_p = 0$ **then**
      $i^\star \in \arg\min_{i \in I_p} \nabla_i f(x^t)$;
    else
      $i^\star \in \arg\min_{i \in I_p^+(x^t)} \nabla_i f(x^t)$;
    end
    $j^\star \in \arg\max_{j \in I_p^+(x^t)} \nabla_j f(x^t)$
    $d^t \leftarrow d_{(p)}^{i^\star, j^\star}$;
    $\beta^t \leftarrow x^t_{(p),j^\star}$;
    compute step α_0^t as in (2.13)
    $\alpha^t \leftarrow QLS(x^t, d^t, \alpha_0^t, \beta^t)$;
    $x^{t+1} \leftarrow x^t + \alpha^t d^t$;
    compute Δ_p^{t+1} as in (2.42)
    if $(t - t_p)$ ***mod*** $L_p = 0$ **then**
      if $\Delta_p^{t+1} > \Delta_p^t$ **then**
        $L_p \leftarrow \max\{L_{min}, L_p/2\}$;
      else
        $L_p \leftarrow \min\{L_{max}, L_p * 2\}$;
      end
      $t_p \leftarrow t$
    end
    $t \leftarrow t + 1$;
  end
end

decreased (see lines 17–24).

Concerning the convergence properties, we observe that Algorithm 3, differently from ISMO, uses different parameters L_p for each O/D pair $p \in P$, whose value may vary during the iterates. Recalling the convergence proof reported in Section 2.4.1, the main assumption is that of ensuring, for each O/D pair, the application of the column generation at least once every $L \cdot |P|$ iterations, where $|P|$ is the number of O/D pairs.

However, as L_p is bounded in $[L_{min}, L_{max}]$ with L_{max} finite, the global convergence properties stated for ISMO still holds by simply replacing in the proof the constant L with L_{max}. Therefore we can state the following result.

Proposition 2.9. *Let $\{x^t\}$ be the sequence generated by Algorithm ISMO-ACG. Then $\{x^t\}$ admits limit points and each limit point is a solution of Problem* (2.6).

Despite the simplicity of the basic ideas underlying the proposed method, its computational performances reported in Section 2.8 are remarkable. Note that the traffic assignment problem has a unique solution in terms of link flows but not in terms of path flows. Then, the proposed approach in ISMO-ACG tries to advantageously exploit the nonuniqueness of the solution driving the iterates towards sparse solutions that may be efficiently computed with limited memory requirements. Indeed, the path-equilibration method, the choice of the initial stepsize in the line search and the strategy of applying the column generation procedure (to possibly generate new variables) only when "necessary", tend to facilitate very sparse solutions. This may improve the rate of convergence to equilibrium points and yields computational advantages to efficiently perform computational operations during the iterates.

2.6 Nonmonotone ISMO-ACG

During the development of ISMO and ISMO-ACG, several solutions, here not reported, have been tested. Most of these were purely technical experiments, while other were methodological, although not supported by a convergence theory. From these, it is worth describing two nonmonotone versions of ISMO-ACG. Both share the abolition of the Quadratic Line Search, thus losing the convergence properties of ISMO and the nonmonotonicity since the Dafermos step, adjusted with respect to the maximum feasible step, is immediately accepted.

The first one is simply the PEA in Algorithm 3 without the line search step, thus the stepsize α^t is given as follows

$$\alpha^t \leftarrow \min\{\alpha_0^t, \beta^t\}. \tag{2.43}$$

The correspondent algorithm, is conveniently named ISMO-NM, where the acronym ACG of ISMO-ACG is removed for the sake of simplicity. In Section 2.8 the results show that the method is able to converge without requiring the convergence properties ensured by the adoption of QLS.

The second version applies a perfect alternation, based on the external number of iterations l, between the Dafermos step and its double,

$$\alpha^t = \begin{cases} \min\{\alpha_0^t, \beta^t\} & \text{if } l \bmod 2 = 0 \\ \min\{2 \cdot \alpha_0^t, \beta^t\} & \text{otherwise.} \end{cases} \tag{2.44}$$

The main motivation of this idea is that, in the first stages of the optimization, several paths with a small amount of flow enter in the working set and remain until the equilibrium is reached.

Accepting a step that is two times the one obtained through the minimization of the quadratic approximation as in Dafermos, the probability of removing these paths increases, thus allowing a better exploiting of the sparsity of the solution in the space of path flows. This latter nonmonotone version is named ISMO-NM-A.

2.7 A framework for Path-Based Algorithms

In this section we aim to show that the column generation strategy employed in algorithm ISMO-ACG can be viewed as a general decomposition framework for generic path-based algorithms.

In Algorithm 4, there are four main components:

(i) the Gauss-Seidel decomposition over O/D pairs $p \in P$;

(ii) the adaptive column generation in step 6;

(iii) the solution update in step 8;

(ii) the adaptive column generation frequency update in step 9.

Algorithm 4: Adaptive Column Generation Framework

Data: $x^0 \in \mathcal{F}$, $L_{min}, L_{max} \in \mathbb{N} : 0 < L_{min} < L_{max} < \infty$

1 $t \leftarrow 0$;
2 $L_p \leftarrow L_{min}$, $t_p \leftarrow t$, $\forall\ p \in P$; $I_p^+(x^t) \leftarrow \{i \in I_p : x^t_{(p),i} > 0\}$;
3 **while** x^t *is not a solution* **do**
4 **for** $p \in P$ **do**
5 **if** $(t - t_p)$ ***mod*** $L_p = 0$ **then**
6 $I_p^+(x^t) \leftarrow i^\star \in \arg\min_{i \in I_p} \nabla_i f(x^t)$;
7 **end**
8 $x^{t+1}, \Delta_p^{t+1} \leftarrow EquilibrationAlgorithm(x^t, I_p^+(x^t))$;
9 **if** $(t - t_p)$ ***mod*** $L_p = 0$ **then**
10 **if** $\Delta_p^{t+1} > \Delta_p^t$ **then**
11 $L_p \leftarrow \max\{L_{min}, L_p/2\}$;
12 **else**
13 $L_p \leftarrow \min\{L_{max}, L_p * 2\}$;
14 **end**
15 $t_p \leftarrow t$
16 **end**
17 $t \leftarrow t + 1$;
18 **end**
19 **end**

In step 8, the value Δ_p^{t+1} is returned by the equilibration algorithm along with the updated solution x^{t+1}. This value represents the reduction of the quadratic model obtained during the equilibration and it is used in the adaptive column generation update of the shortest paths computation frequency. Refer to Section 2.5 for more details.

In next paragraphs, three path-based algorithms are briefly described besides the PEA discussed in Section 2.3 and here reported for the sake of completeness.

PEA

The equilibration phase of the Algorithm 4 can be summarized in the following Algorithm 5.

Algorithm 5: Path-Equilibration Algorithm (PEA)

Input: $x^t \in \mathcal{F}$, $I_p^+(x^t) \in I_p$
Output: x^{t+1}, Δ_p^{t+1}

$i^\star \in \arg \min_{i \in I_p^+(x^t)} \nabla_i f(x^t)$;
$j^\star \in \arg \max_{j \in I_p^+(x^t)} \nabla_j f(x^t)$;
$d^t \leftarrow d_{(p)}^{i^\star, j^\star}$;
$\beta^t \leftarrow x^t_{(p),j^\star}$;
compute step α_0^t as in (2.13)
$\alpha^t \leftarrow QLS(x^t, d^t, \alpha_0^t, \beta^t)$;
$x^{t+1} \leftarrow x^t + \alpha^t d^t$;
compute Δ_p^{t+1} as in (2.42)
return x^{t+1}, Δ_p^{t+1}

Gradient Projection Algorithm

As it is summarized in [52], the method called Gradient Projection (GP) proposed in [28] and further developed in [10] consists in shifting flow from all non-shortest paths to the shortest one.

Each shift is computed as in PEAs, i.e., defining a direction with only two nonzeros components. Then, the chosen stepsize is the Dafermos one in (2.13). Differently from the original method, in Algorithm 6 and in the following ones the line search procedure QLS has been introduced in order to perform a fair comparison with respect to the proposed method ISMO-ACG in the computational experiments reported in Section 2.8.

Rosen Gradient Projection Algorithm

In [16] the Gradient Projection Algorithm is developed on the basis of the Rosen's method (see [54]).

Algorithm 6: Gradient Projection (GP) Algorithm

Input: $x^t \in \mathcal{F}$, $I_p^+(x^t) \in I_p$
Output: x^{t+1}, Δ_p^{t+1}

$i^\star \in \arg \min_{i \in I_p^+(x^t)} \nabla_i f(x^t)$;
$\Delta_p^{t+1} \leftarrow 0$;
for $j \in I_p^+(x^t), j \neq i^\star$ **do**
  $d_j \leftarrow d_{(p)}^{i^\star,j}$;
  $\beta_j \leftarrow x_{(p),j}^t$;
  compute step α_j^0 as in (2.13)
  $\alpha_j \leftarrow QLS(x^t, d_j, \alpha_j^0, \beta_j)$;
  $d_j^t \leftarrow -\alpha_j$
  compute Δ_j as in (2.42);
  $\Delta_p^{t+1} \leftarrow \Delta_p^{t+1} + \Delta_j$;
end
$d_{i^\star}^t \leftarrow - \sum_{j \in I_p^+(x^t), j \neq i^\star} d_j^t$
$x^{t+1} \leftarrow x^t + d^t$;
return x^{t+1}, Δ_p^{t+1}

The main idea of the algorithm is to move flow from the paths that have cost greater than the current average path cost $\bar{s}$ to the paths that have cost less than $\bar{s}$. The resulting direction is the following

$$d_j^t \leftarrow \bar{s} - \nabla_i f(x^t), \quad j \in I_p^+(x^t) \tag{2.45}$$

The initial stepsize is computed like in (2.13). However, in this case, the direction is not made of only two nonzero components. We remark that the maximum step β^t is equal to the minimum flow value among paths that have cost greater than the average path cost $\bar{s}$. Then, as shown in Algorithm 7, a line search along d^t is performed and the solution updated.

Incremental Gradient Projection Algorithm

Comparing PEA and GP in Algorithms 5 and 6, it is easy to observe that GP, at each iteration, performs the path equilibration update in PEA, as well

Algorithm 7: Rosen Gradient Projection (RGP) Algorithm

Input: $x^t \in \mathcal{F}$, $I_p^+(x^t) \in I_p$
Output: x^{t+1}, Δ_p^{t+1}

$\bar{s} \leftarrow \sum_{i \in I_p^+(x^t)} \nabla_i f(x^t)/|I_p^+(x^t)|$;
for $i \in I_p^+(x^t)$ **do**
  $d_j^t \leftarrow \bar{s} - \nabla_j f(x^t)$;
end
$\beta^t \leftarrow \min_{j:d_j^t<0} x_{(p),j}^t$;
compute step α_0^t as in (2.13)
$\alpha^t \leftarrow QLS(x^t, d^t, \alpha_0^t, \beta^t)$;
$x^{t+1} \leftarrow x^t + \alpha^t d^t$;
compute Δ_p^{t+1} as in (2.42);
return x^{t+1}, Δ_p^{t+1}

as other equilibrations between the shortest path and all the non-maximum active cost paths. Since only one path, the shortest one, receives flow from the others, an oscillatory behavior may occur when many paths are active at the same time. In RGP this is handled considering the average cost value.

In this version of GP developed in this work, which has been called Incremental Gradient Projection (IGP), subsequent path equilibrations are performed between the shortest path and the non-shortest ones. The Algorithm 8 moves from x^t to x^{t+1} across intermediate solutions $\hat{x}$. In such a way, a monotone descent of the objective function is guaranteed. We remark that the line search procedure QLS described in Appendix A returns a zero step if the direction d^t is not of descent for f. This may occur when path $i^\star$ is no longer the shortest one after one or more equilibrations.

2.8 Computational results

Once the experimental setup is described (test problems, measure of convergence and implementation details, in Sections 2.8.1, 2.8.2 and 2.8.3), the results of algorithms ISMO and ISMO-ACG are reported in 2.8.4. In the following Section 2.8.5, both the convergence capability and the efficiency of

Algorithm 8: Incremental Gradient Projection (IGP) Algorithm

Input: $x^t \in \mathcal{F}$, $I_p^+(x^t) \in I_p$
Output: x^{t+1}, Δ_p^{t+1}

$i^\star \in \arg \min_{i \in I_p^+(x^t)} \nabla_i f(x^t)$;
$\hat{x} \leftarrow x^t$;
$\Delta_p^{t+1} \leftarrow 0$;
for $j \in I_p^+(x^t), j \neq i^\star$ **do**
  $d_j \leftarrow d_{(p)}^{i^\star,j}$;
  $\beta_j \leftarrow x_{(p),j}^t$;
  compute step α_j^0 as in (2.13)
  $\alpha_j \leftarrow QLS(\hat{x}, d_j, \alpha_j^0, \beta_j)$;
  $\hat{x} \leftarrow \hat{x} + \alpha_j d_j$;
  compute Δ_j as in (2.42);
  $\Delta_p^{t+1} \leftarrow \Delta_p^{t+1} + \Delta_j$;
end
$x^{t+1} \leftarrow \hat{x}$;
return x^{t+1}, Δ_p^{t+1}

the nonmonotone versions of ISMO-ACG are evaluated.

In section 2.8.6, we compare the proposed *Path Equilibration Algorithm* ISMO-ACG with other path-based algorithms with the column generation strategy being the same. In other words, each time an O/D pair is selected, the algorithms differ only in the way the direction is defined and the solution is updated, regardless if the shortest paths are computed or not. The goal is to show the effectiveness of sequential minimal optimizations employed in PEAs with respect to the path-based algorithms described in Section 2.7. Furthermore, we will show how the proposed column generation strategy can improve the performances of the other path-based methods.

Finally, in Section 2.8.7, the proposed algorithm ISMO-ACG is compared with state-of-the-art bush-based algorithms. In the last Section 2.8.8, some numerical aspects of ISMO-ACG are analyzed further.

2.8.1 Test problems

The test problems used for the experiments are described in Table 2.1. All the test problems are freely available at web page `http://www.bgu.ac.il/~bargera/tntp/`. The datasets vary from small networks to big ones, with a large number of links, nodes and O/D pairs to equilibrate. It is worth noting that the intra-zonal trip demands are not considered in the count of the number of O/D pairs.

Network	Code	# links	# nodes	# centroids	# O/D pairs
Sioux-Falls	SF	76	24	24	528
Barcelona	B	2,522	1,020	110	7,922
Winnipeg	W	2,535	1,067	154	4,345
Chicago-Sketch	CS	2,950	933	387	93,135
Berlin-Center	BC	28,376	12,981	865	49,688
Philadelphia	P	40,003	13,389	1,525	1,149,795
Chicago-Regional	CR	39,018	12,982	1,790	2,296,227

Table 2.1: Network datasets details

Finally, the network Sioux-Falls is not used to evaluate the effectiveness of the proposed algorithm, since the comparison on such a small network may be influenced by different implementations of the methods and different test environments. It has been used only to evaluate the numerical behavior of ISMO-ACG discussed in Section 2.8.8.

2.8.2 Measure of convergence

The convergence of the algorithms is evaluated using the *relative gap* (rg) measure (see for instance [6, 7]), defined as follows

$$rg(x) = \frac{\nabla f(x)^\top x - \nabla f(x)^\top \hat{x}}{\nabla f(x)^\top x} \tag{2.46}$$

where

$$\hat{x} \in \arg\min_{y \in \mathcal{F}} \nabla f(x)^\top y \tag{2.47}$$

which corresponds essentially to assign all the demand D_p to the shortest path of each O/D pair $p \in P$.

A point $x^\star \in \mathcal{F}$ is a solution if and only if $rg(x^\star) = 0$. An algorithm generating a sequence $\{x^t\}$ terminates whenever $rg(x^t) \leq \epsilon$, where $\epsilon > 0$ defines the required level of accuracy.

2.8.3 Implementation details

The algorithm has been implemented in C++ and the computation of shortest paths is performed by the Dijkstra implementation available in the well-known Boost C++ libraries collection. In the implementation of the algorithm, shortest paths are computed from a given origin to all other nodes instead of calculating, as formally described in the algorithm, the shortest path between each O/D pair.

All the numerical experiments have been executed on an Ubuntu 14.04 environment with an Intel®Xeon 12-Core E5-2430 2.50GHz and 16 GB of RAM.

The arc cost function $s_a(\cdot)$ is the well-known *Bureau of Public Roads* (BPR) function, which models the link travel time, and takes the form

$$s_a(v_a(x)) = t_0 \left(1 + b \left(\frac{v_a(x)}{c_a}\right)^p\right) \tag{2.48}$$

where t_0 is the free flow travel time, c_a is the link capacity and $b, p \in \mathbb{R}$ are arc-dependent parameters. All these values are available in the Bar-Gera's networks.

For what concerns the parameter L, which represents the column generation period, its definition is discussed throughout the next paragraphs. The parameters of the Quadratic Line Search have been set as follows

$$\delta = 0.5 \qquad \gamma = 10^{-8}$$

while the ISMO parameters are the following

$$c = 10^3 \qquad \tau = 0.05.$$

We recall that $c > 0$ contributes to the ISMO initial stepsize based on first-order information (line 13 in Algorithm 2) and $\tau \in (0, 1)$ is the contraction factor of the stepsize computed by the line search procedure that is employed when numerical issues arise.

2.8.4 Numerical results of ISMO and ISMO-ACG

In this section, the benefits of computing shortest paths periodically with period $L > 1$ is shown compared with the case $L = 1$. Then, moving from ISMO to ISMO-ACG, the effect of the initial step as the one employed in [11] is discussed for cases $L = 1$ and $L > 1$, in order to assess the improvements in terms of both precision and efficiency independently from the column generation frequency. Finally, the effect of the adaptive strategy is discussed.

Recalling the ISMO algorithm 2 presented in Section 2.4, we validate the choice of the value L through the comparison of three different cases[1]:

(i) ISMO-L1: the algorithm ISMO with $L = 1$;

(ii) ISMO-L5: the algorithm ISMO with $L = 5$;

(iii) ISMO-L10: the algorithm ISMO with $L = 10$;

In Table 2.2 the computational times needed to reach various level of the convergence measure rg are reported.

From the results we can observe that such column generation strategy is effective compared with the case $L = 1$. As L increases, the algorithm is able to reach higher precision levels in less time. For instance, the case of dataset Philadelphia (P) is impressive: the algorithm ISMO with $L = 10$ is able to reach the rg value of 10^{-8} in an eighth of the time compared to the case $L = 1$.

This is widely confirmed in Table 2.3 where the number of iterations is reported instead of computational times. On Philadelphia, ISMO-L10 reaches the rg value of 10^{-8} in a comparable number of iterations and a significant saving of time is achieved, since the column generation is performed once every ten iterations. While a greater number of iterations is needed to reach 10^{-6} or 10^{-8} of the relative gap measure, which is rather expected since in most of the iterations the shortest paths might not be considered in the equilibration phase, this is not generally true for higher precisions. On Chicago-Sketch (CS) and on Philadelphia (P), dealing with fewer paths may ease the achievement of more accurate precisions in fewer iterations.

Generally, the case with $L = 10$ seems to be the best one in terms of quality and efficiency and is considered as the default one for ISMO.

Summarizing, the results of ISMO with $L > 1$ shows the effectiveness of such column generation strategy. However, it can be noticed from Figure

[1] A similar test has been performed in [12]

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
B	ISMO-L1	0.55	2.35	13.89	17.23	38.10
	ISMO-L5	0.46	1.52	5.38	7.65	25.53
	ISMO-L10	0.49	0.80	4.38	5.36	7.47
W	ISMO-L1	2.19	3.28	5.50	14.17	17.56
	ISMO-L5	0.64	1.16	1.77	5.04	6.02
	ISMO-L10	0.76	1.25	1.88	4.69	5.53
CS	ISMO-L1	2.77	7.80	11.87	86.60	-
	ISMO-L5	1.73	2.17	3.54	4.54	-
	ISMO-L10	1.77	2.04	3.34	3.73	8.64
BC	ISMO-L1	26.11	41.33	146.50	-	-
	ISMO-L5	13.93	18.47	103.24	-	-
	ISMO-L10	12.41	17.01	66.20	-	-
P	ISMO-L1	438.63	2316.42	-	-	-
	ISMO-L5	185.01	512.62	2557.73	-	-
	ISMO-L10	191.45	363.73	757.68	10356.20	-
CR	ISMO-L1	1026.08	2025.27	4664.05	-	-
	ISMO-L5	475.35	921.94	1228.45	-	-
	ISMO-L10	413.27	892.05	1456.52	-	-

Table 2.2: Comparison of several column generation periods L in terms of CPU time (secs.)

2.1 that the convergence behavior is fairly irregular on all the networks. Especially on networks Chicago-Regional and Philadelphia, a sharp step in the convergence pattern is visible when the algorithm attains a precision of nearly 10^{-9}. This is due to the numerical trick described in Section 2.5 that has been employed in order to deal with numerical issues when the line search procedure is no more able to determine a suitable step.

One of the main concerns of the development of ISMO-ACG is the analysis of such convergence problems, which leads to the adoption of an initial stepsize of QLS obtained by exactly solving a quadratic approximation of the function along the current descent direction, which was firstly employed by Dafermos et al. in [11].

In order to assess the benefits of this enhancement and to confirm the effectiveness of the column generation strategy independently of each other,

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
B	ISMO-L1	21	109	638	779	1731
	ISMO-L5	51	181	618	844	3174
	ISMO-L10	94	144	671	809	1084
W	ISMO-L1	88	133	223	573	716
	ISMO-L5	96	165	238	612	738
	ISMO-L10	163	249	361	774	916
CS	ISMO-L1	22	71	110	770	-
	ISMO-L5	54	71	131	167	-
	ISMO-L10	103	125	221	248	561
BC	ISMO-L1	16	26	96	-	-
	ISMO-L5	38	54	332	-	-
	ISMO-L10	63	93	420	-	-
P	ISMO-L1	84	429	-	-	-
	ISMO-L5	135	342	1332	-	-
	ISMO-L10	258	447	822	6678	-
CR	ISMO-L1	138	273	572	-	-
	ISMO-L5	264	495	636	-	-
	ISMO-L10	420	837	1280	-	-

Table 2.3: Comparison of several column generation periods L in terms of number of iterations.

the algorithms

(i) ISMO-L1-D: ISMO with $L = 1$ and the Dafermos step in (2.14);

(ii) ISMO-L10-D: ISMO with $L = 10$ and the Dafermos step in (2.14).

are compared with ISMO-L1 and ISMO-L10, respectively. Computational results can be found in Table 2.4.

In order to understand properly the reported results, firstly, for each dataset, pairs of rows have to be compared independently, in such a way to validate the adoption of the Dafermos step, regardless the use of the column generation strategy or not. Secondly, the addition of the Dafermos step to ISMO with $L = 10$ has to be evaluated.

For what concern the comparison pair by pair, the use of the Dafermos step leads to significant improvements both in terms of achieved preci-

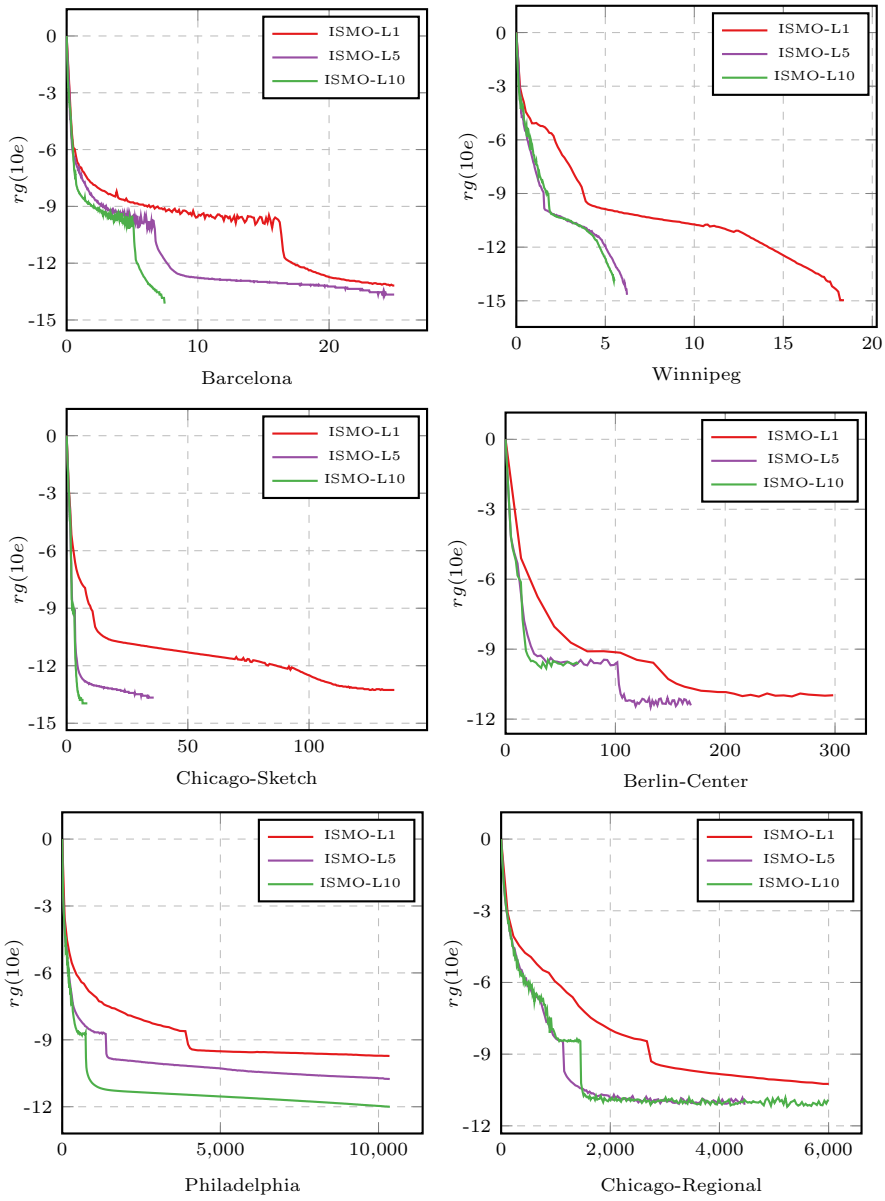


Figure 2.1: Convergence of the algorithms with different column generation periods L. CPU time (s).

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
B	ISMO-L1	0.55	2.35	13.89	17.23	38.10
	ISMO-L1-D	0.45	0.79	1.29	1.87	2.52
	ISMO-L10	0.49	0.80	4.38	5.36	7.47
	ISMO-L10-D	0.30	0.36	0.43	0.5	0.56
W	ISMO-L1	2.19	3.28	5.50	14.17	17.56
	ISMO-L1-D	2.38	3.69	5.02	6.47	7.90
	ISMO-L10	0.76	1.25	1.88	4.69	5.53
	ISMO-L10-D	0.41	0.69	0.99	1.36	1.76
CS	ISMO-L1	2.77	7.80	11.87	86.60	-
	ISMO-L1-D	2.60	6.58	11.40	16.31	20.75
	ISMO-L10	1.77	2.04	3.34	3.73	8.64
	ISMO-L10-D	1.59	1.94	2.20	2.48	2.86
BC	ISMO-L1	26.11	41.33	146.50	-	-
	ISMO-L1-D	22.73	52.64	271.58	280.55	292.53
	ISMO-L10	12.41	17.01	66.20	-	-
	ISMO-L10-D	9.40	14.05	17.09	20.14	23.19
P	ISMO-L1	438.63	2316.42	-	-	-
	ISMO-L1-D	431.34	1700.28	4368.99	6815.69	7665.03
	ISMO-L10	191.45	363.73	757.68	10356.20	-
	ISMO-L10-D	146.00	257.10	533.84	820.32	952.24
CR	ISMO-L1	1026.08	2025.27	4664.05	-	-
	ISMO-L1-D	983.34	1938.45	3451.31	5132.97	5949.00
	ISMO-L10	413.27	892.05	1456.52	-	-
	ISMO-L10-D	327.47	529.65	816.80	1133.21	1337.31

Table 2.4: Comparison of first-order and second-order based stepsize in terms of CPU time (secs.)

sion (10^{-14} in all tests) and computational times. The exception of dataset Berlin-Center (BC) for the case $L = 1$ is not relevant since, as it is discussed later, such dataset has a few characteristics which differ substantially from the ones of the other considered datasets.

From the results, it is clear that the column generation strategy and the nature of the employed initial step affect the resulting equilibrium solutions independently of each other. The test ISMO-L10-D, i.e., ISMO with the

Dafermos step, shows the best performances, with a relevant save of computational time for each dataset along with a high quality solution of the equilibrium, previously unreachable by the algorithm ISMO-L10 (from now on we will get back to the original notation, i.e. ISMO).

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
B	ISMO-L1	21	109	638	779	1731
	ISMO-L1-D	20	38	64	95	133
	ISMO-L10	94	144	671	809	1084
	ISMO-L10-D	93	115	135	169	204
W	ISMO-L1	88	133	223	573	716
	ISMO-L1-D	100	157	216	279	342
	ISMO-L10	163	249	361	774	916
	ISMO-L10-D	100	160	224	306	391
CS	ISMO-L1	22	71	110	770	-
	ISMO-L1-D	21	60	107	155	198
	ISMO-L10	103	125	221	248	561
	ISMO-L10-D	90	112	132	155	184
BC	ISMO-L1	16	26	96	-	-
	ISMO-L1-D	14	34	180	186	194
	ISMO-L10	63	93	420	-	-
	ISMO-L10-D	42	72	100	117	133
P	ISMO-L1	84	429	-	-	-
	ISMO-L1-D	84	330	826	1282	1440
	ISMO-L10	258	447	822	6678	-
	ISMO-L10-D	207	345	636	942	1098
CR	ISMO-L1	138	273	572	-	-
	ISMO-L1-D	132	261	464	688	798
	ISMO-L10	420	837	1280	-	-
	ISMO-L10-D	327	510	738	972	1134

Table 2.5: Comparison of first-order and second-order based stepsize in terms of number of iterations.

This efficiency, however, can not be attributed to a computational complexity difference of the two ways in which the initial stepsize is computed. Indeed, in Table 2.5, such computational gap is clearly justified by the sig-

nificant difference of the number of iterations needed to reach various accuracies. For example, on Philadelphia (P), ISMO-L10 requires 6678 iterations to reach rg equal to 10^{-12}, while ISMO-L10-D reaches needs only 942 iterations.

In Figure 2.2 the convergence patterns are reported. Again, two pairs of lines can be discerned, with or without the employing of the column generation. Nevertheless, the adoption of Dafermos step results in a smooth convergence of the algorithm compared to the case of the use of first-order information for the initial step. In this way, the finite representation of real numbers is suitably handled by the algorithm without requiring any trick.

Finally, for what concerns this series of comparisons, starting from a generic PEA (as ISMO with $L = 1$ is) and ending with the proposed method ISMO-ACG, we present the computational results of algorithm ISMO-ACG in Table 2.6, where only the test ISMO-L10-D is considered. We recall that ISMO-ACG is the ISMO algorithm with the initial stepsize described in (2.14) and with the adaptive column generation strategy described in Section 2.5. Regarding the parameters of the adaptive column generation strategy, L_{min} has been set equal to $L = 10$, while L_{max}, whose definition is less critical as it has only to be finite in order to ensure convergence, has been set equal to 120. In this case, the convergence plots of ISMO-ACG as well as the table which summarize the number of iterations are not reported, since the behavior is comparable with the one of ISMO-L10-D on all networks.

We can observe that the adoption of the adaptive column generation strategy yields computational advantages particularly when high levels of precision are required. For instance, in the Philadelphia test, the two versions are comparable to reach a rg equal to 10^{-6}, while ISMO-ACG requires about half of the CPU time spent by ISMO-L10-D to reach a value of rg equal to 10^{-14}.

The adoption of the Dafermos stepsize allows accurate updates of the solution and this affect positively the quality of the equilibrium and the required time to reach a given precision, since the oscillatory behavior is avoided. The column generation strategy introduced in ISMO and the improved one in ISMO-ACG lead to significant time savings. However, reducing and differentiating the frequency of the shortest paths computation by itself is not enough to motivate these relevant results. An increasing number of iterations needed to reach a given precision may be expected, indeed. Thus, we may obtain comparable computational times with respect to the case in

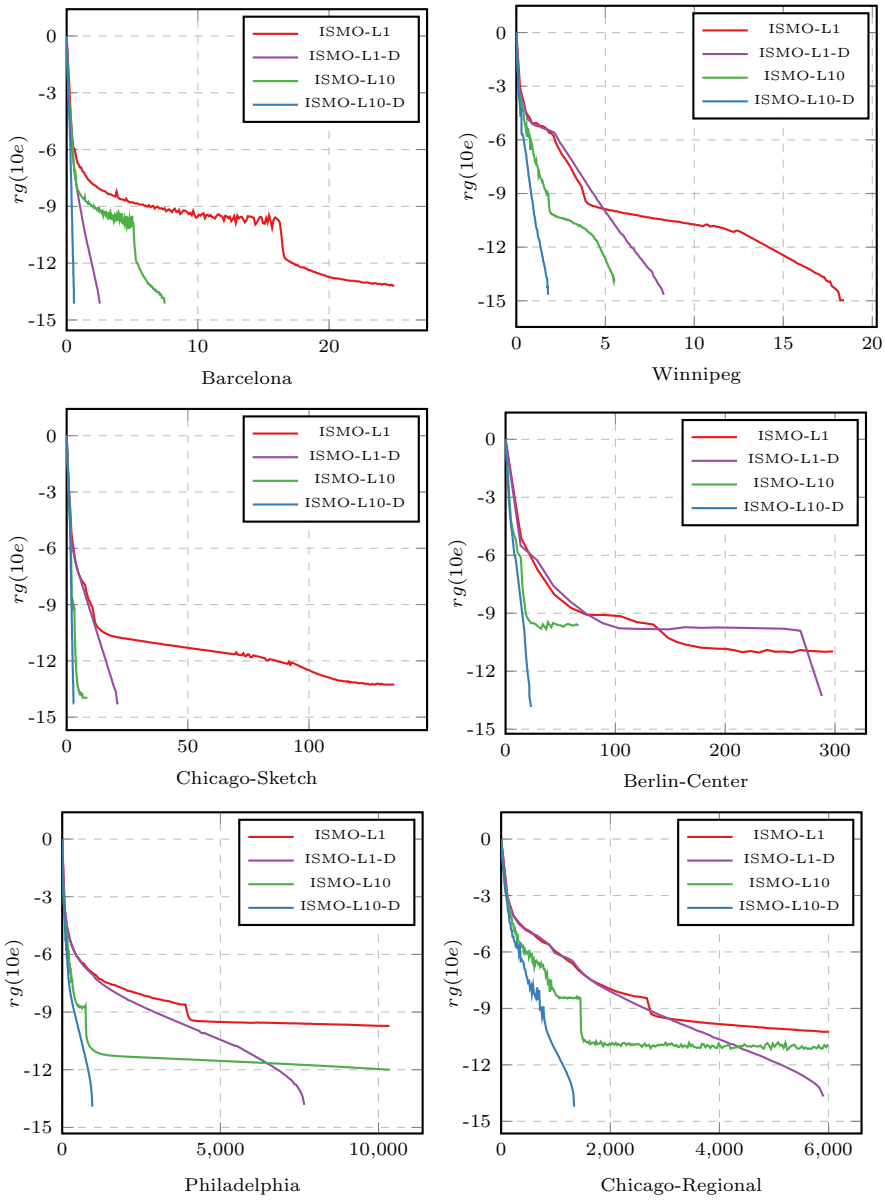


Figure 2.2: Comparison of the convergence behavior of algorithms with the Dafermos step. CPU time (s).

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
B	ISMO-L10-D	0.30	0.36	0.43	0.50	0.56
	ISMO-ACG	0.29	0.38	0.44	0.51	0.56
W	ISMO-L10-D	0.41	0.69	0.99	1.36	1.76
	ISMO-ACG	0.30	0.50	0.62	0.85	1.11
CS	ISMO-L10-D	1.59	1.94	2.20	2.48	2.86
	ISMO-ACG	1.55	1.83	1.90	2.07	2.19
BC	ISMO-L10-D	9.40	14.05	17.09	20.14	23.19
	ISMO-ACG	7.27	7.91	9.09	9.91	10.10
P	ISMO-L10-D	146.00	257.10	533.84	820.32	952.24
	ISMO-ACG	136.02	211.51	353.74	481.83	538.98
CR	ISMO-L10-D	327.47	529.65	816.80	1133.21	1337.31
	ISMO-ACG	313.71	500.29	646.33	881.91	981.67

Table 2.6: Computational results of the introduction of the adaptive column generation. CPU time (secs.)

which no column generation strategy is employed.

Anyway, this is not the case. The proposed algorithms are able to reach a given precision requiring a comparable number of iterations and thus requiring less paths to be equilibrated, since shortest path are not computed at each iteration. Recalling that the traffic assignment problem has a unique solution in terms of link flows but not in terms of path flows, the sparsity of the solution may be exploited in order to obtain an equilibrium with the minimum possible number of involved active paths, since less paths to be equilibrated means less computational time.

The numbers reported in Table 2.7 for the two largest networks confirm what has been remarked, where the Dafermos step has been considered. Less paths are required by ISMO-L10-D and ISMO-ACG to reach a given precision of the solution, indeed. This is easily deducible from the percentages of O/D pairs with at least two (active) paths at different values of the relative gap.

We note that, as we may expect, in the case of ISMO-L1-D such a percentage strongly increases with the number of iterations since the column generation is performed at each iteration and new paths are possibly added.

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
P	ISMO-L1-D	25.88 %	39.36 %	47.76 %	49.82 %	50.08 %
	ISMO-L10-D	10.44 %	18.35 %	25.84 %	28.38 %	28.60 %
	ISMO-ACG	10.85 %	16.75 %	21.32 %	22.77 %	22.94 %
CR	ISMO-L1-D	21.44 %	27.86 %	33.09 %	36.23 %	37.14 %
	ISMO-L10-D	9.28 %	13.49 %	17.92 %	21.32 %	21.93 %
	ISMO-ACG	9.08 %	13.01 %	15.77 %	18.03 %	18.22 %

Table 2.7: Percentage of O/D pairs with at least two (active) paths at the given precision.

With the column generation strategy employed in ISMO-L10-D, such percentage is more or less halved. Thus, the solution obtained has many more O/D pairs with only one active path than that of ISMO-L1-D and this yields advantages in terms of overall effort. Indeed, the O/D pairs having only one active path are skipped by the minimization process and this leads to obvious computational advantages.

ISMO-ACG, compared to ISMO-L10-D, is able to further reduce such percentage, exploiting better the sparsity of the solution. The adaptive strategy has the effect of generating less paths since the parameter L_p is adjusted according to a measure of effectiveness of the column generation, as it is described in Algorithm 3, and new paths are inserted less often through iterations. This is widely confirmed in Figures 2.3 and 2.4, where the mean value and the final distribution of L_p are reported, respectively, for the two largest networks Chicago-Regional and Philadelphia.

The mean value of L_p computed on all O/D pairs starts from $L = 10$ and grows as the algorithm converges to the solution, confirming that the column generation is gradually and globally less useful. In fact, in Figure 2.4 we can observe that only a small percentage of O/D pairs at the end of the optimization has required to generate new paths with the initial highest frequency of $L = 10$.

2.8.5 Numerical results of nonmonotone ISMO-ACG

The nonmonotone methods ISMO-NM and ISMO-NM-A described in Section 2.6 are evaluated with respect to the globally convergent algorithm ISMO-ACG. Two different aspects are of interest in this experiment:

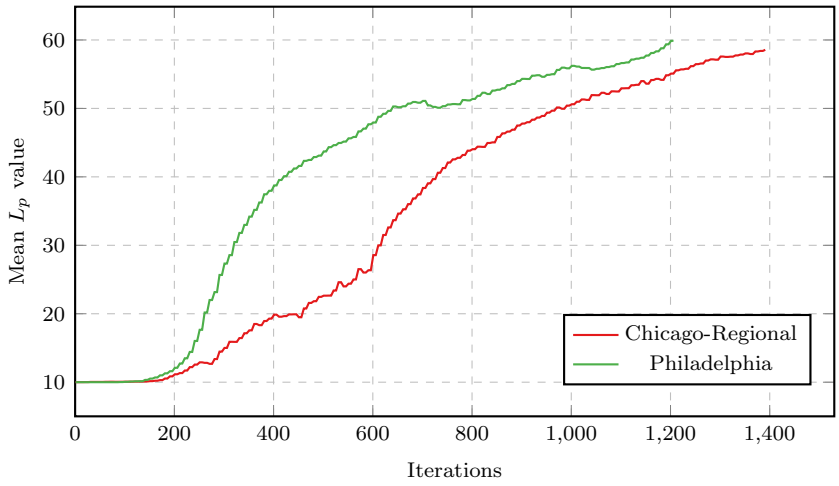


Figure 2.3: Mean L_p parameter on all O/D pairs through iterations.

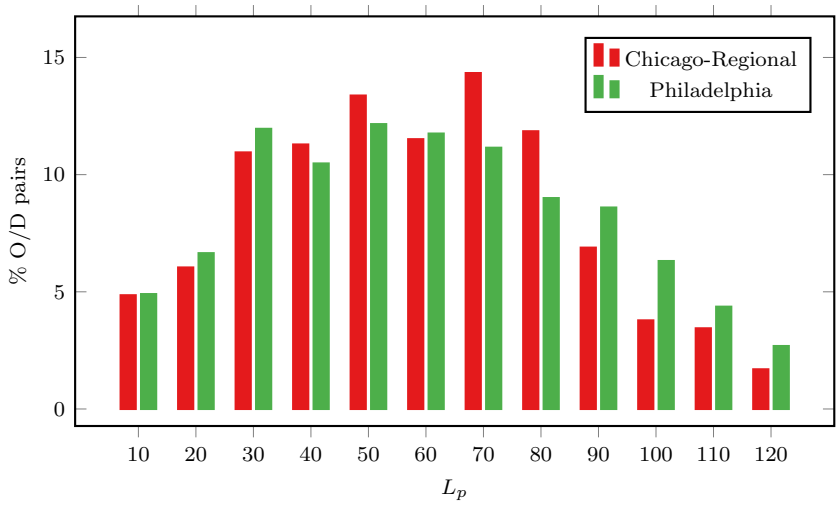


Figure 2.4: Final distribution of L_p parameter.

(i) whether or not the nonmonotone algorithms converge;

(ii) whether or not computational advantages exist removing QLS and exploiting the sparsity of the solution;

We recall that the nonmonotone methods ISMO-NM and ISMO-NM-A share with ISMO-ACG the same adaptive column generation strategy.

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
B	ISMO-ACG	0.29	0.38	0.44	0.51	0.56
	ISMO-NM	0.27	0.53	0.58	0.61	0.62
	ISMO-NM-A	0.30	0.35	0.37	0.40	0.42
W	ISMO-ACG	0.30	0.50	0.62	0.85	1.11
	ISMO-NM	0.32	0.40	0.55	0.70	0.80
	ISMO-NM-A	0.44	0.51	0.67	0.74	0.82
CS	ISMO-ACG	1.55	1.83	1.90	2.07	2.19
	ISMO-NM	1.59	1.88	1.97	2.15	2.34
	ISMO-NM-A	1.74	1.93	2.08	2.30	2.56
BC	ISMO-ACG	7.27	7.91	9.09	9.91	10.10
	ISMO-NM	7.41	8.36	9.34	10.16	10.45
	ISMO-NM-A	7.99	8.57	9.27	9.82	10.13
P	ISMO-ACG	136.02	211.51	353.74	481.83	538.98
	ISMO-NM	132.00	202.86	313.93	427.17	517.62
	ISMO-NM-A	143.75	191.86	265.28	345.66	380.82
CR	ISMO-ACG	313.71	500.29	646.33	881.91	981.67
	ISMO-NM	301.09	444.49	542.18	672.86	-
	ISMO-NM-A	295.17	392.10	429.18	478.45	-

Table 2.8: Results of the nonmonotone algorithms in terms of CPU time (secs.)

In Table 2.8 the results are reported. For what concern the convergence to the solution, ISMO-NM and ISMO-NM-A are able to converge to $rg = 10^{-14}$ on almost all the networks. The only exception is on Chicago-Regional, where $rg = 10^{-12}$ is the maximum reached precision. Although it can be easily interpreted as an effect of nonmonotonicity, actually the reason is purely numerical. In Section 2.8.8 we will show how the way in which the relative gap is computed can impact the reliability of the measure.

In the case of ISMO-NM, the abolition of QLS leads to a computational saving which can be observed in tests Philadelphia and Chicago-Regional, although a general greater number of iterations is required to reach the same precision than ISMO-ACG, as shown in Table 2.9. Shortest paths are the

main computational demanding part of the optimization and the Dafermos step has turned out to generally satisfy the QLS condition without requiring subsequent reductions, thus, explaining such slight differences between ISMO-ACG and ISMO-NM in terms of CPU time.

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
B	ISMO-ACG	91	123	148	178	211
	ISMO-NM	80	270	308	334	355
	ISMO-NM-A	85	119	137	165	183
W	ISMO-ACG	89	169	224	302	389
	ISMO-NM	105	167	254	326	405
	ISMO-NM-A	133	173	241	289	345
CS	ISMO-ACG	84	114	130	154	186
	ISMO-NM	84	112	127	151	182
	ISMO-NM-A	99	117	137	171	227
BC	ISMO-ACG	52	64	92	112	128
	ISMO-NM	52	72	100	116	131
	ISMO-NM-A	64	82	100	106	112
P	ISMO-ACG	207	342	636	944	1104
	ISMO-NM	207	342	620	914	1174
	ISMO-NM-A	237	423	801	1191	1383
CR	ISMO-ACG	327	549	740	1092	1270
	ISMO-NM	327	519	686	928	-
	ISMO-NM-A	375	717	873	1077	-

Table 2.9: Results of the nonmonotone algorithms in terms of number of iterations.

The case of ISMO-NM-A is instead of interest. It is worth observing that a direct comparison with ISMO-ACG on CPU time should not be performed, since ISMO-NM-A does not employ the line search procedure in order to carry out the alternation described in Section 2.6. Thus, it is evaluated with respect to ISMO-NM. The alternation of the Dafermos step and its double lets a significant reduction of the computational times on the two largest networks despite a greater number of iterations needed by the algorithm.

The reason of the efficiency of ISMO-NM-A lies in the evolution of the set of active paths for each O/D pair through iterations, reported in Table

2.10. In such table, ISMO-NM is not reported since it shows a behavior similar to ISMO-ACG, while ISMO-L1-D and ISMO-L10-D are reported for the sake of completeness and in order to let the reader figure out how the sparsity of the equilibrium solution in terms of path flows can be exploited with the relative gap being equal.

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
P	ISMO-L1-D	25.88 %	39.36 %	47.76 %	49.82 %	50.08 %
	ISMO-L10-D	10.44 %	18.35 %	25.84 %	28.38 %	28.60 %
	ISMO-ACG	10.85 %	16.75 %	21.32 %	22.77 %	22.94 %
	ISMO-NM-A	3.31 %	6.18 %	9.12 %	10.43 %	10.91 %
CR	ISMO-L1-D	21.44 %	27.86 %	33.09 %	36.23 %	37.14 %
	ISMO-L10-D	9.28 %	13.49 %	17.92 %	21.32 %	21.93 %
	ISMO-ACG	9.08 %	13.01 %	15.77 %	18.03 %	18.22 %
	ISMO-NM-A	2.36 %	3.76 %	5.54 %	6.10 %	-

Table 2.10: Percentage of O/D pairs with at least two (active) paths at the given precision.

The percentages reported in Table 2.10 related to ISMO-NM-A are indeed impressive. Considering the network Philadelphia, in the first stage of the optimization, let's say $rg = 10^{-6}$, ISMO-NM-A requires to equilibrate a percentage of O/D pairs which is more than three times lower than the one of ISMO-ACG and nearly eight times lower than in the case of ISMO-L1-D. The effect of the alternation lets to maintain active a number of paths which is far less than the previous methods. At the end of the optimization on Philadelphia, only 10.91% of O/D pairs have at least two active paths. This means that approximately 90% are in equilibrium with only one active path.

Such advanced exploiting of the sparsity of the equilibrium solution has two important benefits. The first is about the computational advantages that performing fewer equilibration steps causes. The latter is regarding the applicability of the path-based methods on large-scale networks for what concern the requirement of computer memory. It is well-known that path-based methods suffer from the fast growing of the space of paths and that column generation strategies must be employed. If only a small part of the O/D pairs collects more than one active path, the requirement of memory is sustainable even for very large networks.

In conclusion, the convergence patterns of ISMO-NM and ISMO-NM-A

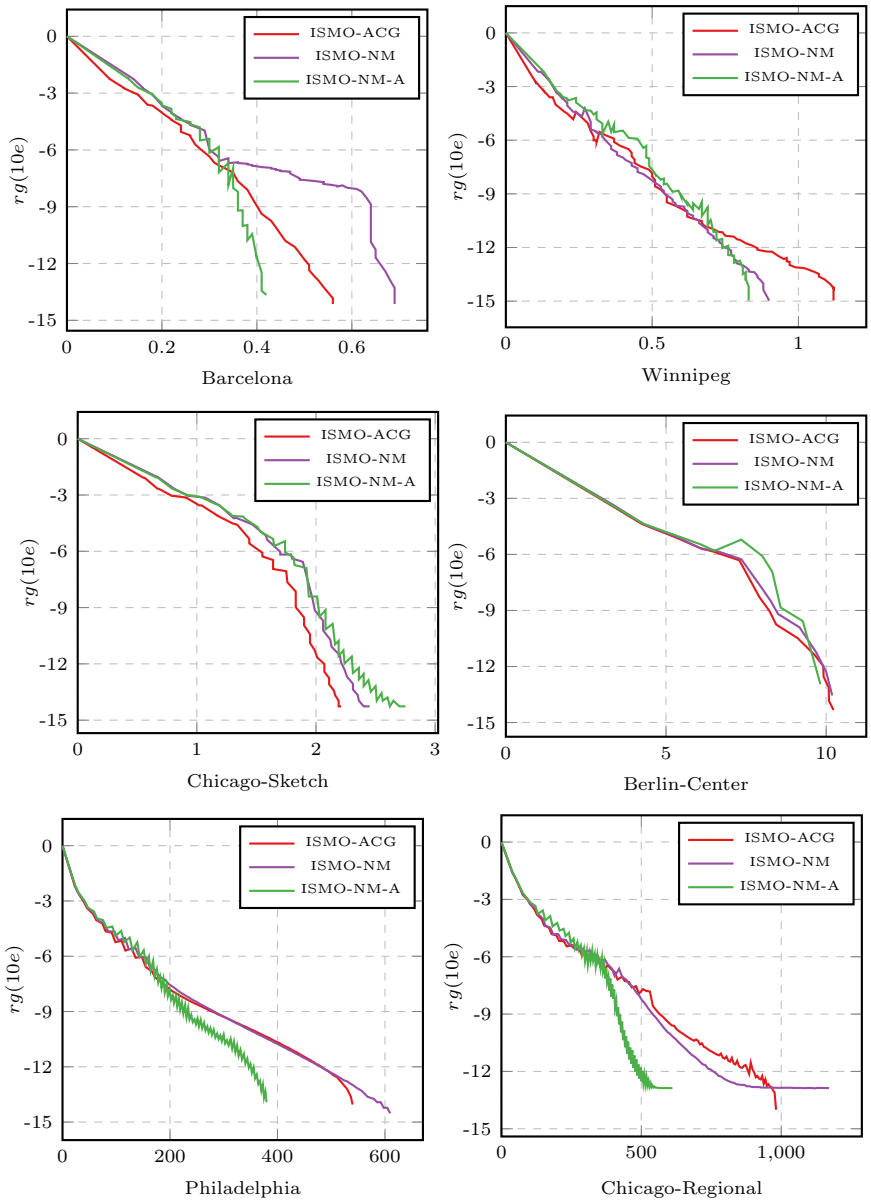


Figure 2.5: Comparison of the convergence behavior of the nonmonotone algorithms. CPU time (s).

are reported in Figure 2.5. While ISMO-NM is comparable with ISMO-ACG, ISMO-NM-A shows the expected oscillatory behavior caused by the alternation process.

Despite the performances of ISMO-NM-A, it has not be considered for the publication yet, mainly due to the lack of convergence properties.

2.8.6 ISMO-ACG compared to other path-based algorithms

In this section we compare ISMO-ACG, which is a PEA, with the three different path-based algorithms described in Section 2.7:

(i) Gradient Projection (GP) algorithm, developed in [28] and [10];

(ii) Rosen Gradient Projection (RGP) algorithm, proposed in [16];

(iii) Incremental Gradient Projection (IGP) algorithm, a monotone variant of GP proposed in this work.

Moreover, we will show how the column generation strategies employed in ISMO and then in ISMO-ACG can improve both the efficiency and the convergence capability of the other path-based methods.

In order to obtain a fair comparison between ISMO-ACG and the other path-based algorithms, the adaptive column generation strategy ACG has been considered.

From Table 2.11 it turns out that the PEA method ISMO-ACG is globally the most efficient. While all the four methods are comparable on datasets Barcelona (B), Winnipeg (W), Chicago-Sketch (CS) and Berlin-Center (BC), on datasets Philadelphia (P) and Chicago-Regional (CR) differences in terms of CPU time are more evident. Generally, RGP-AGC is the less efficient method, while GP-AGC encounters difficulties in reaching high precisions on Chicago-Regional (CR). For what concerns the comparison between GP-ACG and IGP-ACG, the incremental version proposed in this work is clearly better than the original one from both efficiency and effectiveness. Furthermore, IGP-ACG is comparable with ISMO-ACG, although the latter is more efficient in all the cases.

Finally, in Figure 2.6 we report the convergence patterns of GP, RGP and IGP on the two largest networks, Philadelphia and Chicago-Regional, with three different column strategies:

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
B	ISMO-ACG	0.29	0.38	0.44	0.51	0.56
	GP-ACG	0.31	0.39	0.45	0.50	0.55
	RGP-ACG	0.42	0.54	0.65	0.77	0.86
	IGP-ACG	0.31	0.35	0.39	0.45	0.49
W	ISMO-ACG	0.30	0.50	0.62	0.85	1.11
	GP-ACG	0.51	0.86	1.25	1.67	1.96
	RGP-ACG	0.76	1.46	2.28	3.11	3.82
	IGP-ACG	0.38	0.63	0.88	1.13	1.39
CS	ISMO-ACG	1.55	1.83	1.90	2.07	2.19
	GP-ACG	1.55	1.97	2.05	2.22	2.41
	RGP-ACG	1.90	2.18	2.44	2.79	3.17
	IGP-ACG	1.65	1.82	1.95	2.10	2.30
BC	ISMO-ACG	7.27	7.91	9.09	9.91	10.10
	GP-ACG	7.29	8.26	9.20	10.01	10.21
	RGP-ACG	7.59	8.31	8.92	10.14	10.68
	IGP-ACG	7.28	8.25	9.23	10.03	10.36
P	ISMO-ACG	136.02	211.51	353.74	481.83	538.98
	GP-ACG	166.95	290.37	503.15	717.57	814.26
	RGP-ACG	280.39	686.31	1523.70	1950.90	2113.20
	IGP-ACG	149.99	259.80	465.49	635.97	697.47
CR	ISMO-ACG	313.71	500.29	646.33	881.91	981.67
	GP-ACG	340.80	567.05	732.65	-	-
	RGP-ACG	771.89	1338.94	2189.50	2947.13	3568.30
	IGP-ACG	332.38	534.28	697.05	881.25	982.82

Table 2.11: Comparison of several path-based methods in terms of CPU time (secs.)

(i) L1, where shortest paths are computed at each iteration;

(ii) L10, where shortest paths are computed once every $L = 10$ iterations, as in ISMO described in Section 2.4;

(iii) ACG, the adaptive column generation described in Section 2.5.

Starting from the two methods GP and RGP, from Figure 2.6 we can observe that the column generation strategies L10 and ACG are very effective with

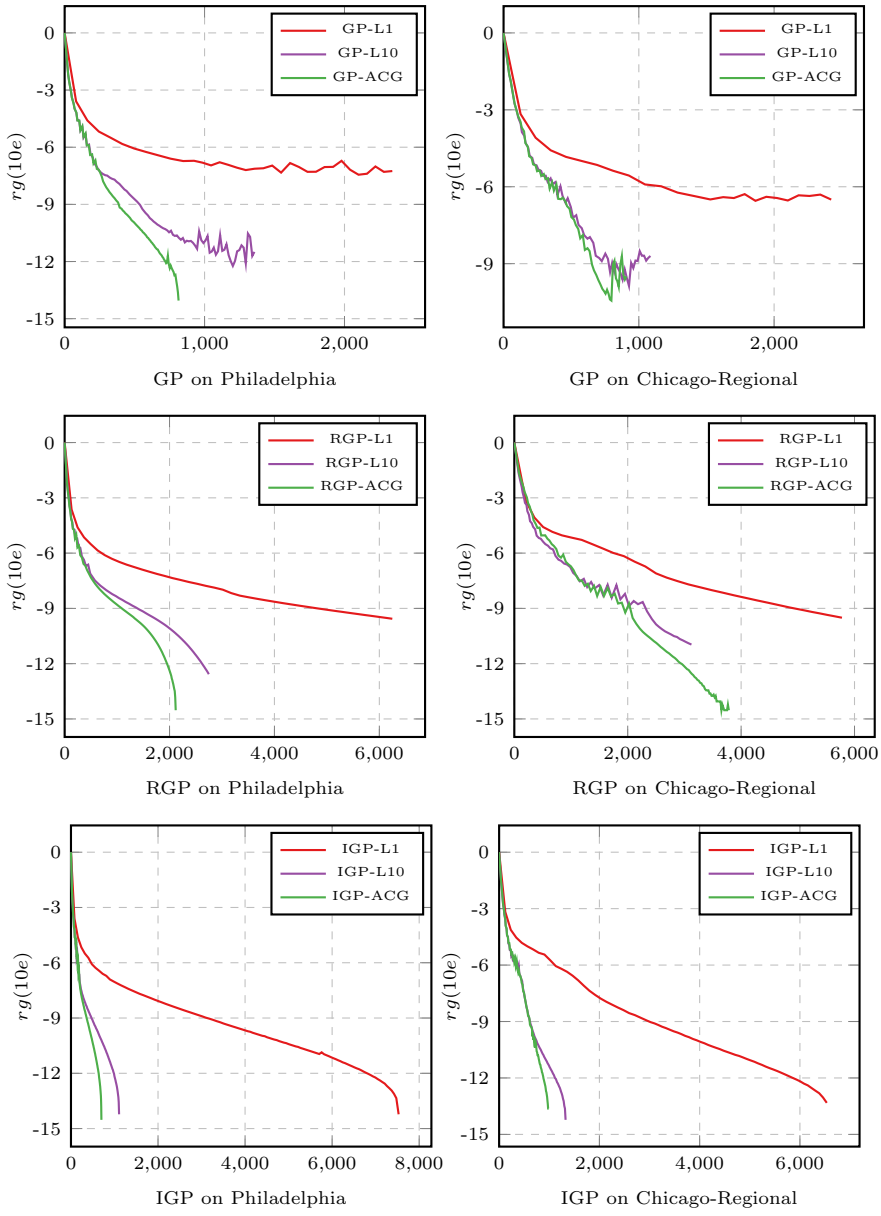


Figure 2.6: GP, RGP and IGP methods with different column generation strategies on Philadelphia and Chicago-Regional. CPU time (s).

respect to L1: computing the shortest paths once every $L = 10$ iterations lets to reach higher precisions in less time, whereas with L1 the two methods GP and RGP stop around a precision of 10^{-6} and 10^{-9} on both networks, respectively. The further specialization of the column generation defined in ACG leads generally to better computational performances and solutions accuracy, especially for RGP.

Although the sparsity of the solution exploited by L10 and ACG could reduce the oscillatory behavior of GP when several non-shortest paths give flow to the shortest one, the convergence patterns of GP on Philadelphia and Chicago-Regional show some oscillations, mainly in the later stages of the optimization.

At the contrary, the method IGP, which has been introduced indeed with the aim of stabilizing the behavior of GP, has a smooth convergence to very high precisions even without any column generation strategies, thus showing a better robustness than GP.

The employment of L10 and ACG in IGP has the effect of reducing progressively the computational times on the two networks.

2.8.7 ISMO-ACG compared to bush-based algorithms

We report in Table 2.12 the results of the comparison between ISMO-ACG and state-of-the-art bush-based methods. These latter are NS-SPT and NS-SFST methods proposed by [64] and the method DBA proposed by [13]. For the network Barcelona (B), results of NS-SPT and NS-SFST are not available.

The method DBA has been implemented by the author following [45]. For the methods proposed by [64] we report the results shown in the cited paper, so the comparison is very rough. Furthermore, the results of the methods NS-SPT and NS-SFST are reported in [64] up to a precision of 10^{-12}. The method DBA is able to attain the precision 10^{-14} only on small networks: on Philadelphia and on Chicago-Regional, the method is not able to reach high precisions since it tends to oscillate around a relative gap of 10^{-12} and 10^{-6}, respectively, during later iterations.

We observe that the ideas underlying the proposed PEA and the method DBA are very similar: a feasible and descent search direction related to the longest and shortest paths is defined and a step along it is performed. However, PEAs define the search direction by the shortest path (either among the used paths or computed by the column generation procedure) and the

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
B	NS-SPT	-	-	-	-	-
	NS-SFST	-	-	-	-	-
	DBA	0.62	0.99	1.44	1.99	2.51
	ISMO-ACG	0.29	0.38	0.44	0.51	0.56
W	NS-SPT	2.60	3.40	4.40	5.30	-
	NS-SFST	4.00	4.40	5.20	6.10	-
	DBA	2.80	4.48	6.20	8.18	10.11
	ISMO-ACG	0.30	0.50	0.62	0.85	1.11
CS	NS-SPT	3.20	3.60	4.40	5.30	-
	NS-SFST	3.90	4.60	5.50	6.40	-
	DBA	2.30	4.24	6.78	9.27	12.28
	ISMO-ACG	1.55	1.83	1.90	2.07	2.19
BC	NS-SPT	72.00	72.00	156.00	162.00	-
	NS-SFST	228.00	246.00	264.00	282.00	-
	DBA	55.69	103.18	375.85	390.98	410.37
	ISMO-ACG	7.27	7.91	9.09	9.91	10.10
P	NS-SPT	480.00	708.00	1140.00	1698.00	-
	NS-SFST	540.00	792.00	1266.00	1782.00	-
	DBA	618.48	2515.76	5390.68	8360.53	-
	ISMO-ACG	136.02	211.51	353.74	481.83	538.98
CR	NS-SPT	1278.00	1890.00	3534.00	5418.00	-
	NS-SFST	762.00	1236.00	2412.00	4206.00	-
	DBA	1353.41	-	-	-	-
	ISMO-ACG	313.71	500.29	646.33	881.91	981.67

Table 2.12: Comparison with state-of-the-art algorithms in terms of CPU time (secs.)

maximum cost used path. DBA, as bush-based method, defines the search direction using the longest path and the shortest path within a given bush (i.e., a directed sub-network of the original network, rooted at a given origin, connected and acyclic). As a consequence, the search directions computed by a PEA and DBA are quite different, and this leads to different realizations and performances.

Beyond the ability of reaching high precisions of the equilibrium solutions,

the algorithm ISMO-ACG outperforms the other methods on all networks and for all levels of precision, especially on the medium-big networks of Berlin-Center, Chicago-Regional and Philadelphia.

The case of Berlin-Center is of interest in this analysis, since this network has a few properties which can guide the comprehension of the effectiveness of ISMO-ACG compared to state-of-the-art methods. From Table 2.1 it can be seen that Berlin-Center is a network comparable with Philadelphia and Chicago-Regional in terms of size, but the number of O/D pairs is significantly lower. Thus, as ISMO-ACG is a PEA performing a Gauss-Seidel decomposition over the different trips, the effort of the equilibration of the O/D pairs is far less the effort of computing the shortest paths on the network. As it can be seen in Tables 2.4 and 2.6 as well, the adoption of the column generation strategy employed in ISMO and its refinement in ISMO-ACG leads to a significant saving of computational time.

On Philadelphia and Chicago-Regional the proposed method is very effective with respect to state-of-the-art methods, however the computational gap is less evident than in the case of Berlin-Center, mainly due to a higher number of O/D pairs to equilibrate.

2.8.8 ISMO-ACG and finite numerical precision

In this section we aim to investigate further on the numerical behavior of the proposed algorithm. Firstly, we will show how the computation of the relative gap can be affected by the way it is computed. Lastly, we will analyze the convergence capability of ISMO-ACG beyond the limits of the common double-precision floating point representation of real numbers.

Recalling the *relative gap* measure reported in Section 2.8.2, we derive two equivalent ways to compute it, respectively referenced as *path-based rg* and *arc-based rg*.

Path-based rg_ϕ

With respect to the notation introduced in Section 2.1, we have

$$rg(x) = \frac{\nabla f(x)^\top x - \nabla f(x)^\top \hat{x}}{\nabla f(x)^\top x} = \frac{\sum_{p\in P}\sum_{k\in K_p} s_k(x)\cdot x_k - \sum_{p\in P} s_p^\star D_p}{\sum_{p\in P}\sum_{k\in K_p} s_k(x)\cdot x_k} =$$

$$= \frac{\sum_{p\in P}\left(\sum_{k\in K_p} s_k(x)\cdot x_k - \sum_{k\in K_p} s_p^\star x_k\right)}{\sum_{p\in P}\sum_{k\in K_p} s_k(x)\cdot x_k}$$

where $s_p^\star$ is the shortest path cost of O/D pair $p \in P$. Finally, we have

$$rg_\phi(x) = \frac{\sum_{p\in P}\sum_{k\in K_p} x_k \cdot \left(s_k(x) - s_p^\star\right)}{\sum_{p\in P}\sum_{k\in K_p} s_k(x)\cdot x_k} \tag{2.49}$$

Arc-based rg_α

Recalling that $s_k(x) = \sum_{a\in A}\delta_{ak}s_a(v_a(x))$ and $v_a(x) = \sum_{k\in K}\delta_{ak}x_k$, we have

$$rg(x) = \frac{\nabla f(x)^\top x - \nabla f(x)^\top \hat{x}}{\nabla f(x)^\top x} = \frac{\sum_{k\in K} s_k(x)\cdot x_k - \sum_{p\in P} s_p^\star D_p}{\sum_{k\in K} s_k(x)\cdot x_k} =$$
$$= \frac{\sum_{k\in K}\sum_{a\in A}\delta_{ak}\cdot s_a(v_a(x))\cdot x_k - \sum_{p\in P} s_p^\star D_p}{\sum_{k\in K}\sum_{a\in A}\delta_{ak}\cdot s_a(v_a(x))\cdot x_k} =$$
$$= \frac{\sum_{a\in A} s_a(v_a(x))\cdot\sum_{k\in K}\delta_{ak}\cdot x_k - \sum_{p\in P} s_p^\star D_p}{\sum_{a\in A} s_a(v_a(x))\cdot\sum_{k\in K}\delta_{ak}\cdot x_k}$$

and finally

$$rg_\alpha(x) = \frac{\sum_{a\in A} s_a(v_a(x))\cdot v_a(x) - \sum_{p\in P} s_p^\star D_p}{\sum_{a\in A} s_a(v_a(x))\cdot v_a(x)} = 1 - \frac{\sum_{p\in P} s_p^\star D_p}{\sum_{a\in A} s_a(v_a(x))\cdot v_a(x)} \tag{2.50}$$

Although are mathematically equivalent, the two formulas show a different numerical behavior with finite precision and with respect to the employment TAP algorithm. Since ISMO-ACG is a path-based method and x represents the vector of path flows, the arc flow $v_a(x)$, given as

$$v_a(x) = \sum_{k \in K} \delta_{ak} x_k, \tag{2.51}$$

is the flow summation over the paths that pass through $a \in A$. Depending on the number of flows that contribute to the sum and on the current precision of the equilibrium solution, the error propagation of the sum may affect significantly the reliability of the measure computed as in (2.50). This is generally true even for the path cost $s_k(x)$

$$s_k(x) = \sum_{a \in A} \delta_{ak} s_a(v_a(x)) \tag{2.52}$$

since it is again a summation and $s_a(\cdot)$ relies on a possibly noisy value $v_a(x)$. However, in (2.49), the contribute $\left(s_k(x) - s_p^\star\right)$ is strictly related to the choice of the descent direction in ISMO-ACG. Thus, when no descent direction can be found, the O/D pair is in equilibrium. In other words, there is a direct correspondence between the direction definition and the measurement of the equilibrium given in (2.49). Conversely, this is not the case for the *relative gap* formula in (2.50).

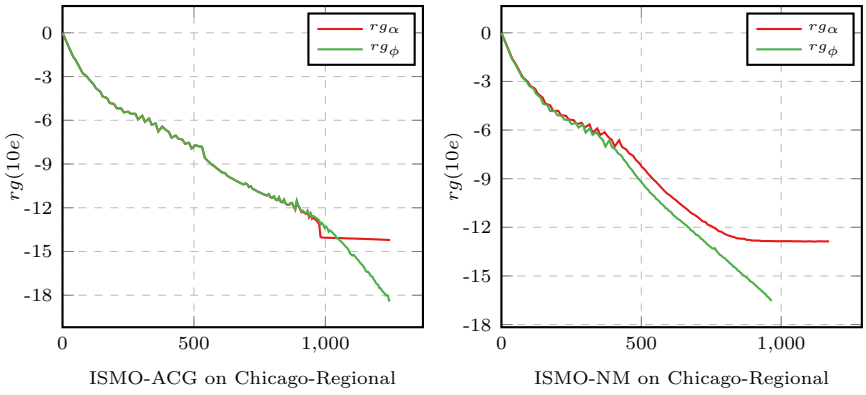


Figure 2.7: Comparison of the convergence behavior of rg_α and rg_ϕ. CPU time (s).

In Figure 2.7 we show an example of the different behavior of rg_α and rg_ϕ on the network Chicago-Regional. In ISMO-ACG, rg_α stops at a precision of 10^{-14}, while the algorithm continues to refine the equilibrium solution since rg_ϕ goes down to 10^{-18}. The case of ISMO-NM is interesting: while in Section 2.8.5 we could observe that the nonmonotonicity might be the reason of a not optimal convergence to high precisions, in Figure 2.7 we can notice that the reason is purely numerical. With rg_ϕ, ISMO-NM is able to reach a precision 10^{-16}, while with rg_α it stops to 10^{-12}.

In conclusion of this numerical analysis, we observe that *path-based* methods should use rg_ϕ as the *relative gap* measure formula, while rg_α is more suitable for *bush-based* algorithms.

Finally, in order to assess the finite precision behavior of ISMO-ACG, we performed further experiments on the smallest network, that is, Sioux-Falls. In particular, we tested ISMO-ACG using three floating point representations available in C++ and reported in Table 2.13.

type	# bits	ϵ	C++ library
float	32	$2^{-24} \approx 5.96 \cdot 10^{-08}$	native
double	64	$2^{-53} \approx 1.11 \cdot 10^{-16}$	native
__float128	128	$2^{-113} \approx 9.63 \cdot 10^{-35}$	quadmath

Table 2.13: Floating point representations

For each floating point representation we plot in Figure 2.8 the relative gap versus the number of iterations (since the quadruple precision is software-simulated and hence the comparison in terms of CPU time is not suitable[2]).

Figure 2.8 shows how the finite precision arithmetic affects the performance of algorithm ISMO-ACG in terms of the level of accuracy of the equilibrium solution. We can observe that ISMO-ACG is able to converge to equilibrium solutions at a precision comparable to the limits of the adopted floating point representation.

[2]As an example, in this test an iteration with 128 bits of precision is nearly 12 times slower than an iteration with 64 bits.

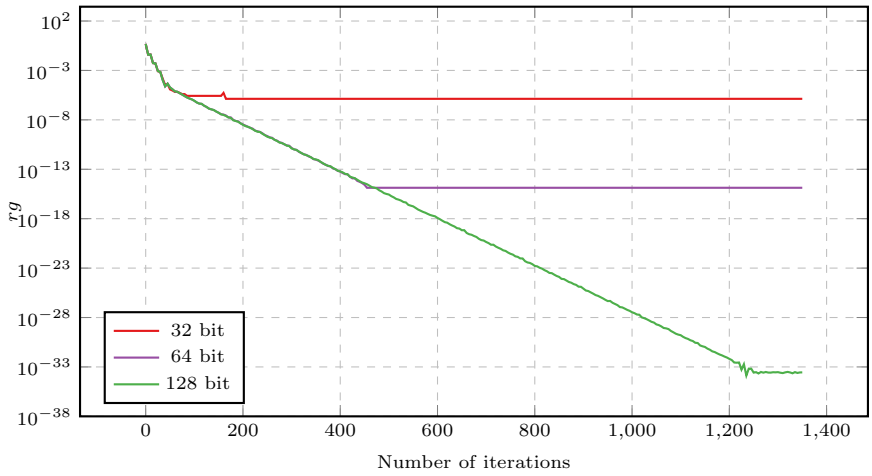


Figure 2.8: The rg measure with the three different floating point representations according to the number of iterations on the Sioux-Falls network

Chapter 3

A computational study on the Elastic Demand Traffic Assignment Problem

While the most standard traffic assignment models assume fixed demand, a more realistic case considers *elastic demands*, that is, the case in which the demand of each O/D pair $p \in P$ depends on the cheapest route cost.

Once the formulation of the equilibrium with fixed demand is defined in Section 3.1, the well-known result of equivalence between inelastic and elastic demand approach is described in Section 3.2. Such result lets the employment of the algorithms developed for the inelastic case even in the elastic case.

We will focus here on a computational study of the path-based algorithms embedded in the general column generation framework described in Section 2.7. Exploiting the equivalence and with minor technical modifications to existing algorithms, we present the numerical results in Section 3.3 which demonstrate the efficiency of such path-based methods in attaining very high levels of precision (comparable with the machine precision) in all the tested networks. To our knowledge, similar levels of precision were never attained by algorithms previously presented in the literature for the case of elastic demand.

3.1 Formulation

For the notation, we generally refer to the one adopted in Section 2.1. For what concern the relation between the demand D_p and the cheapest route cost u_p

$$D_p = \Gamma_p(u_p) \tag{3.1}$$

we assume that it is continuously differentiable, nonnegative, upper bounded, and strictly decreasing.

In order to add the elastic factor in the equilibrium formulation, the inverse function $\Gamma_p^{-1}(\cdot)$ is considered.

$$u_p = \Gamma_p^{-1}(D_p) \tag{3.2}$$

Since $\Gamma_p^{-1}(\cdot)$ is a strictly decreasing and continuosly differentiable function, in the formulation of the TAP with elastic demand, the opposite of $\Gamma_p^{-1}(\cdot)$ will be considered.

Proposition 3.1 (TAP with elastic demand)**.** *With the assumptions stated in Section 2.1, the TAP formulation with elastic demands is given as follows*

$$\begin{aligned} \min_{x,D} \quad & f(x) - g(D) = \sum_{a \in A} \int_0^{v_a(x)} s_a(t)\, dt - \sum_{p \in P} \int_0^{D_p} \Gamma_p^{-1}(w)\, dw \\ s.t. \quad & \sum_{i=1}^{n_p} x_{(p),i} = D_p, \qquad \forall\ p \in P \\ & x_{(p)} \geq 0, \qquad \forall\ p \in P \\ & 0 \leq D_p \leq \Delta_p, \qquad \forall\ p \in P \end{aligned} \tag{3.3}$$

As in Section 2.1, a compact form of 3.3 can be derived

$$\begin{aligned} \min_{x,D} \quad & f(x) - g(D) \\ s.t. \quad & x \in \mathcal{F}(D) = \prod_{p \in P} \mathcal{F}_p(D) \end{aligned} \tag{3.4}$$

where $\mathcal{F}(D)$ is the cartesian product of simplices $\mathcal{F}_p(D)$ defined as follows

$$\mathcal{F}_p(D) = \{x_{(p)} \in \mathbb{R}^{n_p} : \sum_{i=1}^{n_p} x_{(p),i} = D_p,\ x_{(p)} \geq 0\} \tag{3.5}$$

with $0 \leq D_p \leq \Delta_p$.

3.2 On the equivalence between the inelastic and elastic TAP

The algorithms proposed for the inelastic formulation stated in Section 2.1 can be easily extended to the more general case of problem (3.4) with variable demand. The presence of demand variables D with upper bounds can be easily managed by simple network transformations and the problem can be transformed into an equivalent inelastic problem like (2.6).

In this case, we exploit one of the transformations described in [58], the so-called *excess demand formulation*, in which the *excess demand* variable z_p is considered for each O/D pair $p \in P$.

$$z_p = \Delta_p - D_p, \tag{3.6}$$

The variable z_p represents the exceding quantity of demand with respect to the bound Δ_p. Performing the variable substitution in $\Gamma_p^{-1}(\cdot)$ we can define the following function

$$W_p(z_p) = \Gamma_p^{-1}(\Delta_p - z_p) = -\frac{1}{\gamma} \log \frac{\Delta_p - z_p}{\Delta_p} \tag{3.7}$$

and the function $g(\cdot)$ in (3.3), by simple manipulations, can be replaced by the following

$$\begin{aligned} \tilde{g}(z) = g(\Delta - z) &= -\sum_{p \in P} \int_0^{\Delta_p - z_p} \Gamma_p^{-1}(D_p - v)\, d(D_p - v) = \\ &= \sum_{p \in P} \int_{\Delta_p}^{\Delta_p - z_p} W_p(v)\, dv = \\ &= -\sum_{p \in P} \int_{\Delta_p - z_p}^{\Delta_p} W_p(v)\, dv = \\ &= \sum_{p \in P} \int_0^{z_p} W_p(v)\, dv - \sum_{p \in P} \int_0^{\Delta_p} W_p(v)\, dv = \end{aligned} \tag{3.8}$$

Since the second integral is constant with respect to z_p, in the minimization function of (3.3) will be not considered. Thus, we have

$$h(z) = \sum_{p \in P} \int_0^{z_p} W_p(v)\, dv \tag{3.9}$$

and the resultant formulation is the following one.

Proposition 3.2 (Excess demand formulation)**.** *The TAP formulation with elastic demands and excess demand variables is given as follows*

$$\begin{aligned} \min_{x,z} \quad & f(x) - h(z) = \sum_{a \in A} \int_0^{v_a(x)} s_a(t)\, dt + \sum_{p \in P} \int_0^{z_p} W_p(v)\, dv \\ s.t. \quad & \sum_{i=1}^{n_p} x_{(p),i} + z_p = \Delta_p, \qquad \forall\ p \in P \\ & x_{(p)} \geq 0, \qquad \forall\ p \in P \\ & z_p \geq 0 \qquad \forall\ p \in P \end{aligned} \tag{3.10}$$

It is worth noting that for the variable z_p, which is originally bounded in $[0, \Delta_p]$, only the nonnegativity constraint has to be added to the formulation, since it can't be greater than Δ_p due to the demand conservation constraint.

The modelistic interpretation of the formulation (3.10) can be easily derived. The *excess demand* variables add new arcs in the network with cost function $W_p(\cdot)$ and flow value z_p, each one connecting directly an O/D pair $p \in P$. As a consequence, a new path, made only of one arc, enters in the set of paths K_p connecting $p \in P$. An illustration of the transformation is reported in Figure 3.1.

The formulation with *excess demand* variables is indeed a formulation with fixed demands, since Δ_p is fixed. Thus, all the algorithms described in Chapter 2 can be used even in the elastic case.

3.3 Computational Results

The aim of the work described in this chapter is to perform a computational study on the elastic demand traffic assignment problem. In this study, all the path-based algorithms described in Section 2.7 are compared by exploiting the *excess demand* formulation.

In [55] a modification of the gradient projection algorithm, originally proposed in [28] for inelastic problems, has been defined for the elastic case, which is essentially equivalent to the method GP-ACG described in Section 2.7 without any column generation strategy, as the ones employed in ISMO and ISMO-ACG.

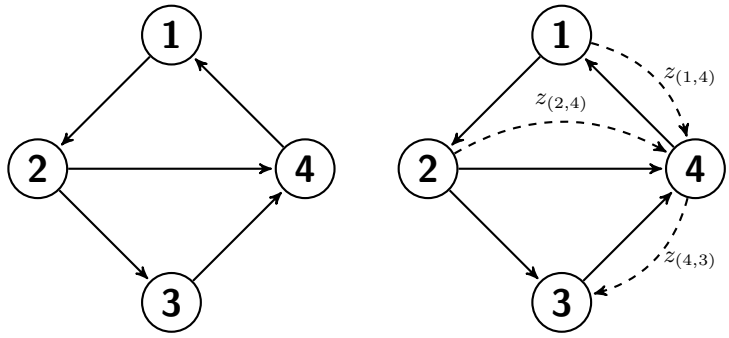


Figure 3.1: a) In the example, the O/D set is composed by three pairs, $(1, 4)$, $(2, 4)$, and $(4, 3)$. b) The addition of excess demand variables leads to a new graph with extra arcs connecting the O/D pairs.

3.3.1 Test problems

The test problems used for the experiments are actually the same as in Section 2.8.1 and are conveniently reported in Table 3.1. However, the network Berlin-Center has been removed since unsolvable numerical problems related to the properties of the network and to the choice of the function $\Gamma(\cdot)$ have been experienced. In its place, a very large-scale network, Sydney, available at `http://www.bgu.ac.il/~bargera/tntp/`, has been added.

Network	Code	# links	# nodes	# centroids	# O/D pairs
Barcelona	B	2,522	1,020	110	7,922
Winnipeg	W	2,535	1,067	154	4,345
Chicago-Sketch	CS	2,950	933	387	93,135
Philadelphia	P	40,003	13,389	1,525	1,149,795
Chicago-Regional	CR	39,018	12,982	1,790	2,296,227
Sydney	S	75,379	33,113	3,264	3,340,619

Table 3.1: Network datasets details

As we will observe in the numerical results, the computational times are greater than in the case of inelastic demand, with the relative gap being equal. This is mainly due to the existence, at each iteration, of at least two

active paths for each O/D pair $p \in P$, one real and the other representing the average excess demand variable, which is generally active, as it is remarked in the implementation details in Section 3.3.3. Thus, at each iteration, each O/D pair is likely to be selected for an equilibration update between at least two paths. The network Sydney has been introduced in order to show that path-based methods can be used on very large networks, even without the exploitation of the solution sparsity discussed in Section 2.8.5.

3.3.2 Measure of convergence

Since a result of equivalence holds, the *relative gap* measure reported in Section 2.8.2 is used even in the elastic case.

$$rg(x) = \frac{\nabla f(x)^\top x - \nabla f(x)^\top \hat{x}}{\nabla f(x)^\top x} \tag{3.11}$$

However, in [55], a different relative gap measure[1] rg_τ is used by the authors

$$rg_\tau(x,z) = \frac{\sum_{p\in P}\sum_{i=1}^{n_p} x_{(p),i} \cdot \left|c_{(p),i} - W_p(z_p)\right|}{\sum_{p\in P}\sum_{i=1}^{n_p} x_{(p),i} \cdot c_{(p),i}} \tag{3.12}$$

where

$$c_{(p),i} = \nabla_{x_{(p),i}} f(x). \tag{3.13}$$

is the cost of i-th route connecting OD pair $p \in P$. However, the employment of this measure may lead to misleading results compared to the standard *relative gap* measure. In Section 3.3.5, a comparison of the two measures is reported.

3.3.3 Implementation details

The implementation details of the path-based algorithms are the ones described in Section 2.8.3.

For what concerns the transformation of the network caused by the addition of the *excess demand* variables, no extra arcs have been actually added

[1] The formula reported in (3.12) is the correct one, as confirmed by the authors of [55]

to the network. The new variables have been maintained exclusively in the set of active paths of each O/D pair as special paths. Its cost is then considered in the objective function.

As in [55], we adopt the following O/D demand function,

$$D_p = \Gamma_p(u_p) = \Delta_p e^{-\gamma u_p}, \qquad \forall\, p \in P \tag{3.14}$$

where Δ_p is an upper bound of the demand D_p, u_p is the least cost path connecting $p \in P$ and $\gamma > 0$ is a factor which reflects the sensitivity of the transport demand with respect to the transport cost. The upper bound Δ_p has been set as double the amount of the original O/D demand provided by the network datasets, while γ has been set to 0.05.

The starting point (x^0, z^0) has been chosen as follows:

- z^0 has been set, for each O/D pair, equal to $\Delta_p/2$, that is, the original O/D demand D_p;
- x^0 has been set by assigning, for each O/D pair, the remaining flow quantity $\Delta_p/2$ to the shortest path in the unloaded network.

In conclusion, it is worth observing that the *excess demand* path with flow z_p is generally active. Its cost function $W_p(\cdot)$ in (3.10) is given as follows

$$W_p(z_p) = -\frac{1}{\gamma} \log \frac{\Delta_p - z_p}{\Delta_p} \tag{3.15}$$

If $z_p = 0$, then the cost $W_p(z_p)$ is equal to zero. Since at least one real path k is active with flow x_k equal to $\Delta_p > 0$, its cost $s_k(x)$ must be zero. However, since $s_a(\cdot)$ is a nonnegative strictly increasing function, as assumed in Assumption 2.3, this can not happen.

Moreover, z_p should not be the only active path for $p \in P$. In such case, $z_p = \Delta_p$ and its cost will be infinite, thus leading to numerical problems when the directional derivative has to be computed. For this reason, a precaution check has been implemented to avoid assigning all the demand to z_p.

3.3.4 Numerical results

The analysis of the numerical results here reported aims to evaluate firstly the convergence capability of the method *Incremental Gradient Projection* (IGP) described in Section 2.7 with respect to the original GP (see [10, 28])

and then to compare all the path-based method reported in Section 2.7 within the *Adaptive Column Generation* framework.

In order to evaluate IGP with respect to GP, in Table 3.2 a comparison without any advanced column generation strategy is performed. Then, in Table 3.3 where all path-based methods are reported, a comparison between IGP and GP with the *Adaptive Column Generation* can be extracted. We recall that, in Section 2.8.4, the solution without column generation has been denoted by the suffix L1, thus in Table 3.2 we will compare IGP-L1 and GP-L1. Furthermore, it is worth noting that GP-L1 corresponds to the *Gradient Projection path-based* method adapted for the elastic Traffic Assignment Problem proposed in [55].

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
B	GP-L1	3.15	6.35	11.36	-	-
	IGP-L1	2.91	6.44	12.25	15.17	18.46
W	GP-L1	4.84	-	-	-	-
	IGP-L1	4.43	10.37	18.75	26.50	33.26
CS	GP-L1	4.02	8.15	12.06	15.58	18.88
	IGP-L1	4.03	7.77	12.92	17.48	22.01
P	GP-L1	640.52	-	-	-	-
	IGP-L1	508.09	2169.64	6973.41	-	-
CR	GP-L1	905.24	-	-	-	-
	IGP-L1	760.69	3368.17	12217.30	-	-
S	GP-L1	1102.28	1401.97	2569.08	-	-
	IGP-L1	1112.65	1751.07	2761.97	-	-

Table 3.2: Comparison between IGP-L1 and GP-L1 in terms of CPU time (secs.)

From Table 3.2, we can observe that the convergence capability of IGP is generally improved with respect to GP. Considering the case of Winnipeg (W), IGP is able to obtain a solution that is eight orders of magnitude more accurate. However, keeping fixed the solution precision, GP is generally more performant.

Introducing the *Adaptive Column Generation* ACG, in Table 3.3 we can observe that the method GP-ACG seems to take advantage of the generally

lower number of existing active paths, as it is discussed in Section 2.8.4 and shown in Table 2.7. Both GP-ACG and IGP-ACG are able to reach the highest precision $rg = 10^{-14}$ on all networks, with the exception of Phladelphia (P) and Chicago-Regional (CR), where IGP-ACG is less accurate then GP-ACG on the first and more accurate on the latter, respectively. Thus, conversely from the case of inelastic demand, IGP-ACG does not lead to a significant improvement of the convergence capability in the case of elastic demand.

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
B	ISMO-ACG	4.23	6.11	8.58	11.29	13.76
	GP-ACG	4.70	7.62	10.08	12.95	14.49
	RGP-ACG	68.08	-	-	-	-
	IGP-ACG	4.75	9.85	12.85	15.14	16.78
W	ISMO-ACG	2.60	4.59	7.20	9.61	11.40
	GP-ACG	3.19	7.57	12.30	16.89	20.50
	RGP-ACG	28.46	-	-	-	-
	IGP-ACG	3.18	5.74	8.53	11.16	13.13
CS	ISMO-ACG	5.40	7.15	10.82	14.01	16.49
	GP-ACG	6.06	8.57	10.66	13.03	14.75
	RGP-ACG	74.73	230.78	-	-	-
	IGP-ACG	7.05	10.27	13.72	16.69	18.93
P	ISMO-ACG	319.52	892.31	2063.07	3125.33	3708.94
	GP-ACG	402.51	1252.01	2972.87	4966.50	-
	RGP-ACG	6634.81	-	-	-	-
	IGP-ACG	365.31	1012.12	-	-	-
CR	ISMO-ACG	663.26	1453.85	2850.60	6204.40	9225.00
	GP-ACG	732.44	2466.67	-	-	-
	RGP-ACG	13253.30	-	-	-	-
	IGP-ACG	753.70	1438.05	6093.89	-	-
S	ISMO-ACG	1374.08	3358.54	3730.85	4530.50	4819.55
	GP-ACG	1264.55	2302.03	3924.95	4850.31	5348.12
	RGP-ACG	-	-	-	-	-
	IGP-ACG	1605.45	2113.11	4438.80	4795.76	5368.34

Table 3.3: Comparison of the path-based methods in terms of CPU time (secs.)

From what concern ISMO-ACG, as it has been already observed in Section 2.8.6, it is the best path-based method among the tested algorithms even in the elastic case. It is able to reach very high precisions on all the networks and it exhibits the best performances with respect to the other methods, with the only exception of network Chicago-Sketch. To our knowledge, such high quality of the solutions were never achieved by existing algorithms in literature. Since ISMO-ACG is the best one, it outperforms the method GP-L1 proposed in [55] as a consequence.

The *Rosen's gradient-based* method RGP-ACG encounters severe difficulties even on small networks, as it can be noticed in Figure 3.2. The reasons behind these results might be represented by either methodological or technical issues. However, at the moment no apparent problem has been found during an analysis of the method execution.

Finally, in Table 3.4, the nonmonotone methods ISMO-NM and ISMO-NM-A described in Section 2.6 are compared with ISMO-ACG.

Since at least two paths are active for each O/D pair, an equilibration is always performed, which involves in algorithm ISMO-ACG the line search procedure QLS described in Appendix A. At the contrary, in ISMO-NM the initial stepsize given by the minimization of the quadratic approximation is immediately accepted, thus leading to a considerable saving of computational time, in the case the convergence is not affected. Except for Chicago-Regional, where ISMO-NM is not able to reach a precision greater than 10^{-8}, this is generally true for the other networks: the performances of ISMO-NM are impressive with respect to ISMO-ACG, especially in the case of networks Philadelphia and Sydney.

Contrary to what has been observed in the inelastic case, ISMO-NM-A is not able to exhibit the same excellent performances even for the elastic case, where the alternation process described in Section 2.6 affects negatively the convergence of the method. Moreover, the sparsity of the solution can't be exploited with the purpose of reducing the number of O/D pairs to equilibrate, since each O/D pair has at least two active paths.

3.3.5 Comparison of rg_ϕ and rg_τ

In this section we will show how the employment of measure $rg_\tau(\cdot)$ may lead to misleading results compared to $rg(\cdot)$. We will consider the formula rg_ϕ derived in (2.49) since rg_τ considers path flows as well. For example, let's consider the following case. Given a network with only one O/D pair $p \in P$,

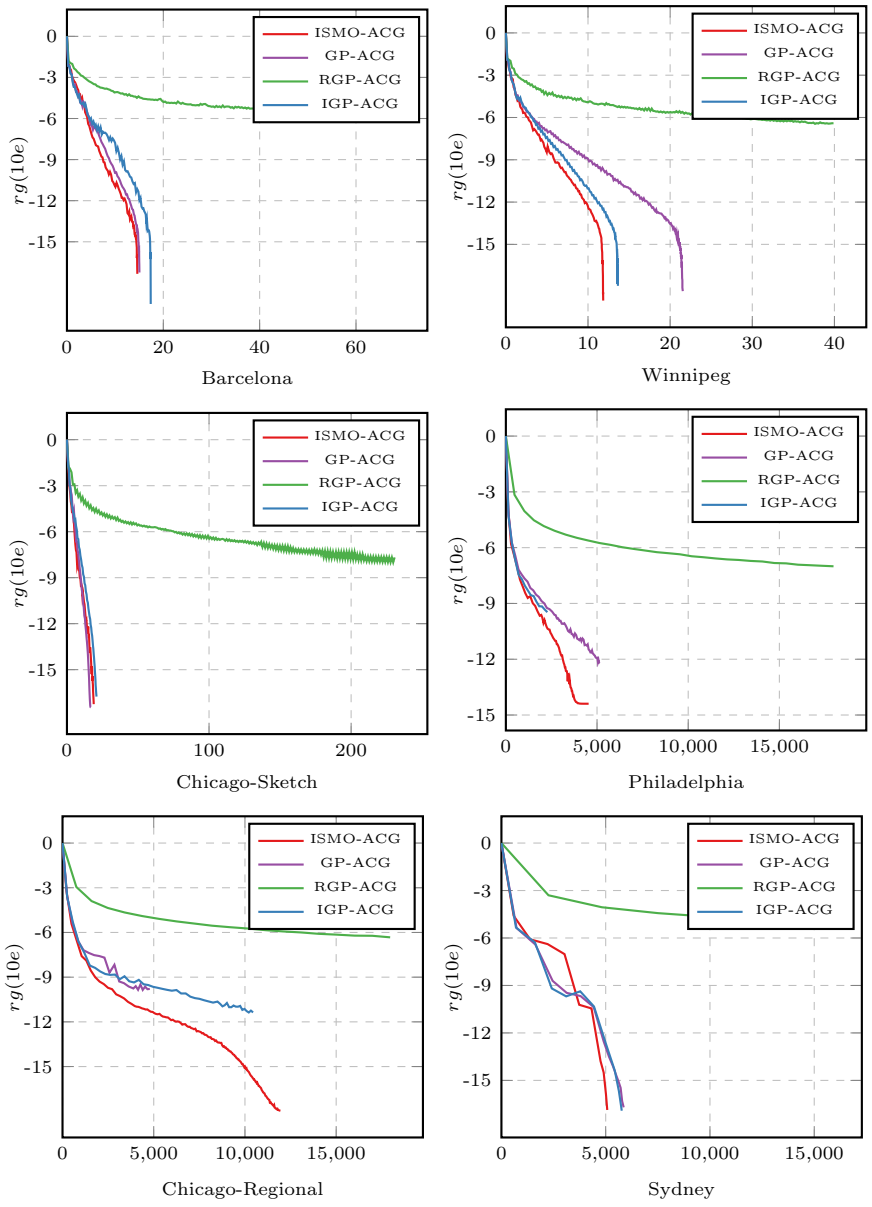


Figure 3.2: Comparison of the convergence behavior of the path-based algorithms. CPU time (s).

Dataset	Algorithm	10^{-6}	10^{-8}	10^{-10}	10^{-12}	10^{-14}
B	ISMO-ACG	4.23	6.11	8.58	11.29	13.76
	ISMO-NM	1.64	2.20	3.11	4.45	5.24
	ISMO-NM-A	3.29	29.39	29.72	30.13	30.95
W	ISMO-ACG	2.60	4.59	7.20	9.61	11.40
	ISMO-NM	1.09	2.06	2.89	3.71	4.46
	ISMO-NM-A	1.54	2.81	3.90	5.21	6.34
CS	ISMO-ACG	5.40	7.15	10.82	14.01	16.49
	ISMO-NM	2.81	3.44	5.11	6.77	7.82
	ISMO-NM-A	7.31	57.39	-	-	-
P	ISMO-ACG	319.52	892.31	2063.07	3125.33	3708.94
	ISMO-NM	157.91	408.59	907.71	1338.05	1673.69
	ISMO-NM-A	320.47	1056.85	-	-	-
CR	ISMO-ACG	663.26	1453.85	2850.60	6204.40	9225.00
	ISMO-NM	302.73	599.14	-	-	-
	ISMO-NM-A	591.83	-	-	-	-
S	ISMO-ACG	1374.08	3358.54	3730.85	4530.50	4819.55
	ISMO-NM	658.24	1338.90	1448.71	1768.95	1885.56
	ISMO-NM-A	3311.78	-	-	-	-

Table 3.4: Results of the nonmonotone algorithms in terms of CPU time (secs.)

the set of active paths is made by the *excess demand* path z_p with cost $W_p(z_p)$, n paths x_k with cost $s_k(x) = W_p(z_p)$ and an other path $x_{k^\star}$, which is the shortest path, with cost $s_{k^\star}(x) = W_p(z_p)/(n+1)$. For simplicity, we will assume that $\xi = z_p = x_k = x_{k^\star} = \Delta_p/(n+2)$, $k = 1, \ldots, n$. Exploiting the *relative gap* formula in (2.49) described in Section 2.8.8, we have

$$rg_\phi(x,z) = \frac{\xi \cdot \sum_{k=1}^{n} (s_k(x) - s_{k^\star}(x)) + \xi \cdot (W_p(z_p) - s_{k^\star}(x))}{\xi \cdot \sum_{i=1}^{n_p} s_k(x) + \xi \cdot W_p(z_p) + \xi \cdot s_{k^\star}(x)} =$$

$$= \frac{(n+1) \cdot (W_p(z_p) - s_{k^\star}(x))}{(n+1) \cdot W_p(z_p) + s_{k^\star}(x)} = \frac{(n+1) \cdot \left(W_p(z_p) - \dfrac{W_p(z_p)}{n+1} \right)}{(n+1) \cdot W_p(z_p) + s_{k^\star}(x)} =$$

$$= \frac{n \cdot W_p(z_p)}{(n+1) \cdot W_p(z_p) + s_{k^\star}(x)}$$

and

$$rg_\tau(x,z) = \frac{\xi \cdot \sum_{k=1}^{n} |s_k(x) - W_p(z_p)| + \xi \cdot |s_{k^\star}(x) - W_p(z_p)|}{\xi \cdot \sum_{i=1}^{n_p} s_k(x) + \xi \cdot W_p(z_p) + \xi \cdot s_{k^\star}(x)} =$$

$$= \frac{|s_{k^\star}(x) - W_p(z_p)|}{(n+1) \cdot W_p(z_p) + s_{k^\star}(x)} = \frac{\left| \frac{W_p(z_p)}{n+1} - W_p(z_p) \right|}{(n+1) \cdot W_p(z_p) + s_{k^\star}(x)} =$$

$$= \frac{\frac{n \cdot W_p(z_p)}{n+1}}{(n+1) \cdot W_p(z_p) + s_{k^\star}(x)}$$

The ratio between the two measures is then

$$\frac{rg_\phi(x,z)}{rg_\tau(x,z)} = n+1 \tag{3.16}$$

This means that, in this case, the true relative gap $rg_\phi(\cdot)$ is $(n+1)$ times the one measured by $rg_\tau(\cdot)$. If $n = 9$, then there is a difference of one order of magnitude between the two measures. Obviously, the O/D pair is not in equilibrium, thus the measure given by $rg_\phi(\cdot)$ is more reliable than the one obtained with $rg_\tau(\cdot)$.

Similar remarks can be done conversely, in case

$$s_k(x) = s_{k^\star}(x) = W_p(z_p)/(n+1).$$

This time,

$$\frac{rg_\phi(x,z)}{rg_\tau(x,z)} = \frac{1}{n+1} \tag{3.17}$$

and the equilibrium solution of the O/D pair is $(n+1)$ times more accurate than what is measured by $rg_\tau(\cdot)$.

In Figure 3.3 an example of the gap between these two measures is reported on Chicago-Sketch and Chicago-Regional with algorithm ISMO-NM. While in the first stages of the optimization rg_ϕ and rg_τ are comparable, the latter ends up with a *relative gap* that is lower than rg_ϕ on Chicago-Sketch and higher on Chicago-Regional.

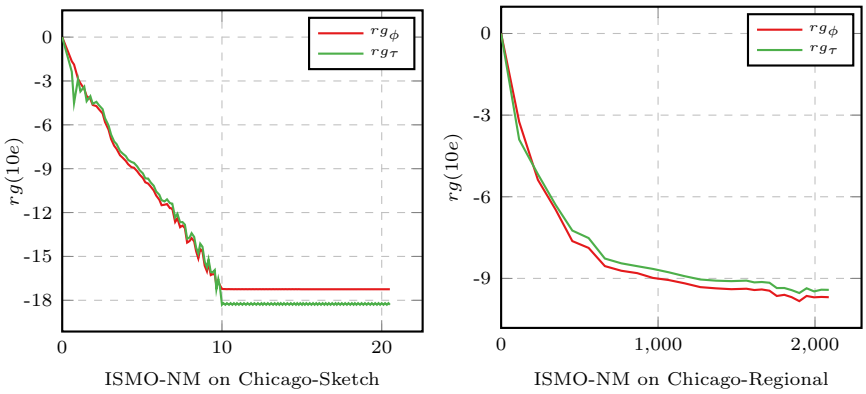


Figure 3.3: Comparison of the convergence behavior of rg_ϕ and rg_τ. CPU time (s).

Finally, we can observe an oscillation behavior of rg_τ right after the achieving of 10^{-3} on Chicago-Sketch, while rg_ϕ shows a smooth convergence and thus seems to be more robust than rg_τ.

Chapter 4

Black-box optimization for O-D Matrix Estimation

In this chapter, the estimation of the Origin/Destination matrix problem is considered. An Origin/Destination (O/D) matrix describes the behavior of the traffic on a network and its knowledge represents an essential input to many traffic models. Transportation demands are a critical requirement, either in static or dynamic models for traffic assignment, indeed.

However, an accurate O/D matrix is rather hard to obtain, since demands are not yet directly observable; thus, the current practice consists of adjusting an initial or a historical matrix from link flow counts, speeds, travel times and other aggregate demand data directly observable.

As smart cities evolve fast, the growing number of sensors lets the collection of useful information for the estimation of O/D matrices of large-scale real networks. The increasing availability of traffic measurements, as it is pointed out in [30], leads to the formulation and development of more accurate and efficient algorithms. However, since most of the existing algorithms rely on the use of a TAP algorithm in a bi-level programming approach, the accuracy of the estimated matrix strongly depends, besides the quality of the seed matrix, on the equilibrium precision achievable by the TAP algorithm. Furthermore, the efficiency of the equilibrium process has a key role in the estimation problem, especially in the case large-scale networks are considered.

In this chapter, the ability of the algorithm ISMO-ACG to reach high precision solutions with high computational performances is exploited in the

O/D matrix estimation problem. The chapter is organized as follows. In Section 4.1, the main contributions on the O/D matrix estimation problem existing in literature are briefly described. In Section 4.2, a basic black-box derivative-free method is presented which takes advantage of the TAP algorithm ISMO-ACG, while in Section 4.3 a decomposition-based approach has been developed in order to deal with real-size network problems. Finally, in Section 4.4, the results of the approaches are discussed, both from the optimization and the estimation points of view.

4.1 Related Work

The problem of estimating an Origin/Destination (O/D) matrix from observed traffic flows on road networks has gained a lot of attention in past decades. Several methods have been proposed and these mainly differ in the way that the observed data is used and the resulting O/D matrix is assigned to the paths of the networks.

Initially, uncongested networks were considered, as in [61]. Then, in [9, 15, 29, 56, 63], congested networks have been introduced in the estimation problem, leading to the formulation of a bi-level programming problem. In such methods, the observed link flows are assumed to be an equilibrium flow pattern with respect to the Wardrop's first principle (*User Equilibrium*, [62]) stated in Definition 1 in Chapter 2.

Due to nonconvexity and nondifferentiability, bi-level programming problems are generally hard to solve. Moreover, the evaluation of the objective function or the gradient vector of the upper level (leader) problem requires the computation of the solution of the lower level (follower) problem.

As a consequence, in recent years, many single-level approaches have been proposed in order to overcome the difficulties caused by the bi-level formulation. In [42] the estimation and the equilibrium steps are merged in a single-level continuous formulation, while in [46], and in a following development [59], the Wardrop's user equilibrium principle is not strictly enforced, thus leading to the involvement of paths which do not satisfies the equilibrium conditions.

In [30], the O/D matrix estimation problem has been formulated as a black-box derivative-free optimization problem where various traffic data types have been used. With a dynamic traffic assignment model as a black-box, several derivative-free methods have been tested.

4.2 A black-box approach

The aim is to estimate the O/D matrix D, i.e., the demand D_p for each O/D pair $p \in P$, starting from a set $\bar{A} \subseteq A$ of network arcs for which the arc flow $\bar{v}_a$, $a \in \bar{A}$, is observed. The observations $\bar{v}_a$ are assumed to be relative to an equilibrium state of the network with respect to the *user equilibrium* principle. Starting from the bi-level programming problem

$$\begin{aligned} \min_{x,v,D} \quad & \frac{1}{2} \sum_{a \in \bar{A}} (v_a(x) - \bar{v}_a)^2 \\ s.t. \quad & D_p > 0, \qquad \forall \ p \in P \\ & x \in \arg\min_z \left\{ f(z) : \sum_{i=1}^{n_p} z_{(p),i} = D_p, z_{(p)} \geq 0, \forall\, p \in P \right\} \end{aligned} \tag{4.1}$$

where x is the solution of the equilibrium problem stated in Proposition 2.5 and $v_a(x) = \sum_{k \in K} \delta_{ak} x_k$, the black-box problem is given as follows, with the notation introduced in Chapter 2,

$$\begin{aligned} \min_{D} \quad & F(D) = \frac{1}{2} \sum_{a \in \bar{A}} (v_a - \bar{v}_a)^2 \\ s.t. \quad & v = \mathrm{TAP}(D) \\ & l_b \leq D_p \leq ub \qquad \forall \ p \in P \end{aligned} \tag{4.2}$$

where

$$\mathrm{TAP} : \mathbb{R}^{|V| \times |V|} \to \mathbb{R}^{|A|}$$

is the black-box function that, given the O/D matrix D, returns the vector of arc flows of the network in equilibrium and l_b, u_b are the lower and upper bounds of D_p, respectively.

The aim of this work is to evaluate the algorithm ISMO-ACG presented in Section 2.5 as the black-box function TAP in (4.2). The black-box algorithm developed is basically a coordinate descent algorithm. We refer with ISMO-CD to this solution in the following. In Algorithm 9 a sketch of the adopted approach is reported where we conveniently denote with Δ_e^t the mesh size along the direction e which corresponds to a specific coordinate, i.e., a specific O/D demand variable.

Algorithm 9: Sketch of ISMO-CD Coordinate Descent Algorithm

Data: An initial matrix D^0, an initial mesh size $\Delta^0 \in \mathbb{R}^{|P|}$

$t \leftarrow 0$;
$E \leftarrow \{e_1, \ldots, e_{|P|}, -e_1, \ldots, -e_{|P|}\}$;
while *a termination criterion is not met* **do**
  $d \leftarrow \arg\min_{e \in E} F(D^t + \Delta^t_e e)$;
  $\bar{D} \leftarrow D^t + \Delta^t_d \, d$;
  if $F(\bar{D}) < F(D^t)$ **then**
    $D^{t+1} \leftarrow \bar{D}$;
  else
    $D^{t+1} \leftarrow D^t$;
  end
  $\Delta^{t+1} \leftarrow \text{updateMesh}(\Delta^t, F(\bar{D}), F(D^t))$;
  $t \leftarrow t + 1$;
end

4.3 A decomposition-based black-box approach

The results reported in the literature generally refer to test networks with limited size and number of O/D pairs, whose demand has to be estimated, mainly due to the difficulties encountered by the existing methods when dealing with large-scale real networks. Here we apply a typical Gauss-Seidel decomposition on the O/D demand variables in order to decompose the problem and take advantage of a smaller dimension of the subproblems.

Similar solution have been employed in [23] and in [37], where derivative information is available only on a subset of blocks, while on the others a derivative-free approach is used.

In Algorithm 10 we report the integration of ISMO-CD in a Gauss-Seidel decomposition framework. The method is called D-ISMO-CD.

4.4 Computational results

The computational results are divided in two main parts. In the first one, the algorithm ISMO-CD is evaluated and then compared with other estimation

Algorithm 10: Decomposition-based ISMO-CD (D-ISMO-CD)

Data: An initial matrix $D^0 = \{D_1^0, D_2^0, \ldots, D_m^0\}$, a maximum number of iterations t_{max}, a block decomposition $B = (b_1, \ldots, b_m)$

```
t ← 0;
while t < t_max do
    for b ∈ B do
        D_b^{t+1} ∈ arg min_ξ F(D_1^{t+1}, D_2^{t+1}, ..., ξ, ..., D_m^t) using ISMO-CD;
    end
    D^{t+1} = (D_1^{t+1}, D_2^{t+1}, ..., D_m^{t+1});
    t ← t + 1;
end
```

methods existing in literature on benchmark networks. In the second part, the decomposition-based solution D-ISMO-CD is compared with ISMO-CD on a larger network then the benchmark ones in order to evaluate the effect of the space decomposition in the estimation process.

For what concern ISMO-CD, in order to perform a robust evaluation of the effectiveness of an O/D estimation algorithm, a validation framework has to be defined. In this work, we refer to the framework reported in [59], which is quite commonly used in the O/D estimation literature.

4.4.1 Test problems

The test problems used for the experiments are described in Table 4.1.

Network	Code	# links	# nodes	# centroids	# O/D pairs
Yang	Y	14	9	4	4
Sioux-Falls Small	SFS	76	24	9	6
Sioux-Falls	SF	76	24	24	528

Table 4.1: Network datasets details

The two small networks are used as benchmark networks for the evaluation of the estimation capability of ISMO-CD whereas the original Sioux-Falls network with 528 O/D demands is used for the tests concerning the decomposition-based approach D-ISMO-CD, since such dimension is com-

monly considered as large-scale for the estimation of the O/D demand matrix.

It can be observed that the first two networks count a very small number of O/D pairs as they are commonly used as a benchmark for O/D estimation algorithm. The network Yang is a small grid originally appeared in [63] and used in [46, 59]. The network Sioux-Falls Small differs from the original network Sioux-Falls in the O/D pairs number. The networks Yang and Sioux-Falls Small are reported in Figures 4.1 and 4.2, along with the true O/D matrices.

4.4.2 Measure of performances

The effectiveness of the algorithm is measured evaluating the quality of the O/D matrix estimation. In [59], several measures have been used. In this work, however, only the Percentage Root Mean Square Error (PRMSE) is used. Denoting with $D^\star$ the true O/D demand matrix, the PRMSE measure is as follows.

$$PRMSE(D) = 100 \cdot \sqrt{\frac{\sum_{p \in P}(D_p - D_p^\star)^2}{|P|}} \Bigg/ \frac{\sum_{p \in P} D_p^\star}{|P|} \tag{4.3}$$

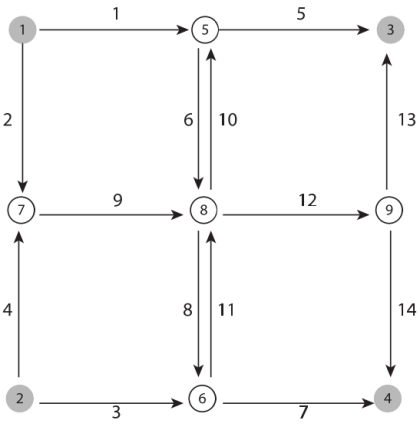
4.4.3 Validation framework

Once a true O/D matrix $D^\star$ is defined, as the ones available in benchmark networks Yang and Sioux-Falls Small, an equilibrium solution is computed and the arc flows on the network in equilibrium, which represents link counts, are considered as the only data source available.

Then, the estimation process is evaluated varying the following parameters.

- Standard deviation σ_p of a normally distributed variable used to sample an initial historical demand $\hat{D}_p$ from the true one $D_p^\star$: since the true matrix $D^\star$ is unknown and has to be estimated, a historical matrix $\hat{D}$ is built, where each O/D demand $\hat{D}_p$ is sampled from a normally distributed variable with mean $D_p^\star$ and variance σ_p^2. The value σ_p has been set proportional to the true demand $D_p^\star$, i.e.,

$$\sigma_p = \epsilon D_p^\star$$

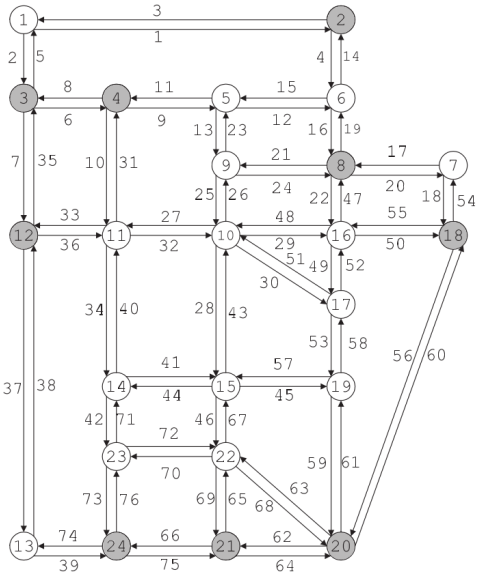


(a) Network

O	D	Demand
1	3	200
1	4	150
2	3	140
2	4	185

(b) True O/D matrix

Figure 4.1: Yang (image by [59])



(a) Network

O	D	Demand
4	20	20000
4	21	10000
12	2	5000
12	18	10000
24	3	15000
24	8	15000

(b) True O/D matrix

Figure 4.2: Sioux-Falls Small (image by [59])

where several values of ϵ has been chosen in order to simulate different levels of reliability of the historical matrix $\hat{D}$,

$$\epsilon \in \{0.01, 0.1, 0.2, \infty\}.$$

With $\epsilon = \infty$ we conveniently indicate the case of maximum uncertainty, in which the historical matrix $\hat{D}$ is set to zero.

- The probability parameter $\bar{p}$ of a Bernoulli distribution such that different coverage cases of link counts can be evaluated[1]. Values chosen for p are the following.

$$\bar{p} \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$$

Besides the evaluation of the quality of the estimation related to different combinations of accuracy of the historical demand matrix and coverage percentage of link counts, the effect of the TAP algorithm has been evalutated with respect to different precisions of the equilibrium solution. Since the assumption on the observed data is that the link flows satisfy the Wardrop's User Equilibrium condition, the quality of the solution of the black-box function may affect the estimation process. For this reason, several precision levels have been considered

$$rg \in \{10e^{-4}, 10e^{-7}, 10e^{-10}, 10e^{-14}\}$$

where rg is the Relative Gap measure described in Section 2.8.2.

Finally, each test combination

$$(\epsilon, \bar{p}, rg)$$

is performed ten times and the mean values of the objective function $F(\cdot)$ and PRMSE measure are considered.

4.4.4 Implementation details

The TAP algorithm used for the O/D estimation is ISMO-ACG with the implementation details described in Section 2.8.3. For what concern the

[1] However, a sensor coverage pattern with parameter $\bar{p}$ does not necessarily have $100 \cdot \bar{p}\%$ of the links being chosen as observed links.

derivative free method of coordinate descent, the NOMAD library [1] inspired by the contributions [3,33] has been employed, which is a C++ black-box optimization software. In order to perform a basic coordinate descent optimization, only the set of directions GPS_2N_STATIC has been used and the model search as well as the opportunistic evaluation and the spectulative search have been disabled. Besides the default termination criteria employed by NOMAD, as the limits of the mesh size for example, for each run a maximum number of evaluations has been set to 5000.

4.4.5 Numerical results of ISMO-CD

In Tables 4.2 and 4.3 are reported the results of ISMO-CD on networks Yang and Sioux-Falls Small concerning the final objective function $F(\cdot)$ of the black-box problem (4.2) and the O/D matrix estimation measure PRMSE, respectively. In order to understand properly the results reported in the tables, a threshold equal to 10^{-5} has been identified and results with the value greater than such threshold are reported in bold face and considered as *failures* of either the optimization of (4.2) or the O/D matrix estimation. In this way, the evaluation of the effect of different values of ϵ, $\bar{p}$ and rg can be discussed without actually analyze the numbers in details.

Starting from the results of the objective function, from Table 4.2 it is clear that the performances improve as the coverage of link count increases. For what concern the reliability of the historical matrix, as the uncertainty around the true demand matrix increases, the algorithm reaches the optimum less frequently. However, the performances do not degrade as expected, even for low percentages of link coverage or for the case $\epsilon = \infty$.

This is generally true for both networks Yang and Sioux-Falls Small. However, in case of low precisions of the TAP algorithm, significant difference in the results can be observed when the parameter ϵ varies from 0.01 to 0.2. When $rg = 10^{-4}$, the poor quality of the equilibrium solution together with a high uncertainty of the historical matrix lead to local minima. From Table 4.2, it can be observed that high precisions of the equilibrium lead to better performances in general, especially in the case of network Sioux-Falls Small. However, since each result is the average value of ten runs, some of them may be affected by individual runs that fail to reach the optimum, leading to a high final average value. As an example, the cases of $(\epsilon = 0.1, \bar{p} = 0.2, rg = 10^{-4})$ and $(\epsilon = 0.1, \bar{p} = 0.2, rg = 10^{-14})$ discredit the idea that high precisions of the TAP algorithm lead to better estimation results.

rg(1e-)	ϵ	$\bar{p}$=0.2	$\bar{p}$=0.3	$\bar{p}$=0.4	$\bar{p}$=0.5	$\bar{p}$=0.6	$\bar{p}$=0.7	$\bar{p}$=0.8	$\bar{p}$=0.9
4	0.01	0.0e+00	7.1e-25	3.7e-26	5.6e-25	**1.0e-04**	3.4e-26	8.6e-26	3.6e-26
4	0.1	2.7e-25	5.7e-27	2.5e-26	2.8e-25	9.0e-26	1.5e-25	4.0e-25	1.6e-25
4	0.2	**1.6e+02**	6.5e-26	3.2e-24	1.0e-25	6.0e-26	9.5e-26	2.6e-25	1.1e-25
4	∞	0.0e+00	2.3e-27	1.3e-28	3.0e-26	0.0e+00	0.0e+00	3.3e-26	1.7e-28
7	0.01	1.3e-25	3.6e-20	5.5e-25	9.9e-23	5.1e-25	1.1e-25	7.1e-26	8.2e-26
7	0.1	0.0e+00	1.5e-25	2.1e-06	9.2e-25	7.8e-27	4.5e-26	8.4e-26	2.0e-26
7	0.2	**3.8e+02**	9.0e-26	3.1e-25	2.3e-07	9.7e-26	6.4e-26	6.0e-26	1.7e-25
7	∞	7.8e-26	0.0e+00	1.4e-25	2.9e-08	3.7e-26	2.9e-26	5.4e-26	5.3e-26
10	0.01	1.2e-24	5.0e-26	1.2e-23	**5.1e-04**	1.5e-21	1.3e-20	1.2e-21	1.2e-21
10	0.1	1.2e-20	**9.6e-05**	**1.3e+00**	3.3e-21	5.6e-20	9.1e-22	9.7e-21	2.5e-21
10	0.2	8.2e-27	3.5e-07	**3.0e+00**	**1.9e-02**	2.9e-21	3.4e-21	4.7e-21	5.7e-21
10	∞	2.2e-17	5.6e-21	2.0e-23	2.0e-21	3.4e-29	2.8e-24	1.7e-21	6.6e-21
14	0.01	3.2e-26	2.6e-25	1.3e-25	3.9e-11	1.4e-23	1.6e-24	6.5e-24	6.1e-24
14	0.1	1.1e-25	4.6e-23	8.7e-24	8.7e-24	1.6e-23	8.8e-24	5.4e-24	2.0e-23
14	0.2	6.8e-22	1.2e-23	1.1e-23	**9.4e-02**	2.4e-24	1.5e-23	2.0e-25	1.7e-23
14	∞	1.6e-25	1.1e-27	7.9e-26	7.9e-24	2.0e-25	1.5e-23	2.0e-23	1.7e-23

(a) Yang

rg(1e-)	ϵ	$\bar{p}$=0.2	$\bar{p}$=0.3	$\bar{p}$=0.4	$\bar{p}$=0.5	$\bar{p}$=0.6	$\bar{p}$=0.7	$\bar{p}$=0.8	$\bar{p}$=0.9
4	0.01	2.8e-20	5.7e-16	1.6e-22	**2.7e+00**	8.3e-23	**2.3e-01**	9.2e-23	8.2e-23
4	0.1	1.0e-22	2.2e-22	2.6e-22	5.0e-22	**2.2e-01**	**3.9e-01**	**3.3e-01**	**2.9e-01**
4	0.2	**4.0e+05**	**9.5e+04**	**5.8e+00**	**4.2e+05**	1.8e-22	1.1e-22	1.5e-22	**3.2e-01**
4	∞	**5.7e-01**	6.1e-23	6.1e-23	9.7e-23	1.1e-22	7.8e-23	2.0e-22	7.8e-23
7	0.01	**8.2e-03**	3.9e-08	2.2e-07	3.6e-08	2.1e-10	4.0e-07	1.3e-07	2.3e-07
7	0.1	**1.5e+00**	2.4e-08	1.9e-07	8.7e-10	1.9e-10	2.2e-09	4.2e-13	1.6e-09
7	0.2	1.1e-07	**2.6e-02**	3.9e-12	4.1e-09	2.2e-09	6.2e-11	3.7e-08	1.0e-13
7	∞	9.8e-09	2.0e-07	7.5e-16	1.1e-08	1.5e-14	1.7e-08	7.7e-16	1.1e-11
10	0.01	1.1e-07	**4.6e-03**	2.8e-12	6.5e-13	1.0e-12	1.5e-12	5.6e-13	8.1e-13
10	0.1	2.7e-12	8.5e-13	7.3e-13	1.1e-12	8.6e-13	9.6e-13	2.2e-13	2.2e-13
10	0.2	3.8e-08	**1.8e-01**	1.2e-12	7.5e-12	1.3e-12	4.7e-13	4.8e-13	5.0e-13
10	∞	**2.3e+05**	**5.2e+00**	1.5e-13	3.8e-13	5.0e-13	1.2e-12	2.1e-13	4.8e-13
14	0.01	**3.0e+00**	**1.5e-02**	2.6e-20	2.2e-20	2.4e-20	6.8e-20	2.1e-20	8.9e-21
14	0.1	**1.3e+04**	9.0e-14	6.9e-20	2.2e-20	2.0e-20	1.5e-20	8.5e-20	2.7e-20
14	0.2	**9.3e+03**	2.0e-20	1.7e-06	1.8e-20	2.5e-20	3.0e-16	4.2e-20	5.3e-18
14	∞	**7.0e+00**	1.3e-20	2.2e-20	4.0e-21	5.4e-21	1.2e-20	8.3e-21	1.3e-20

(b) Sioux-Falls Small

Table 4.2: Final objective function values. Values greater than $1e-5$ are reported in bold face.

Analogous remarks can be done for the analysis of the PRMSE measures reported in Table 4.3. While solving the black-box problem does not imply obtaining a clean estimation of the true O/D demand matrix in general, in this case a correspondence can be found, mainly because the networks are small and only a few O/D pairs are considered. There nevertheless is the case of Yang, in which a good estimation of the true demand matrix seems

rg(1e-)	ϵ	$\bar{p}$=0.2	$\bar{p}$=0.3	$\bar{p}$=0.4	$\bar{p}$=0.5	$\bar{p}$=0.6	$\bar{p}$=0.7	$\bar{p}$=0.8	$\bar{p}$=0.9
4	0.01	**1.1e+01**	1.3e-12	**5.9e-01**	1.7e-12	**9.0e-01**	6.2e-13	2.4e-13	1.4e-13
4	0.1	**7.4e+00**	**5.2e+00**	7.5e-13	1.4e-12	9.6e-13	3.7e-13	1.0e-12	4.0e-13
4	0.2	**4.7e+01**	1.7e-13	9.2e-12	2.4e-13	1.7e-13	3.0e-13	3.6e-13	2.6e-13
4	∞	0.0e+00	**1.1e+02**	0.0e+00	1.3e-13	0.0e+00	0.0e+00	1.3e-13	0.0e+00
7	0.01	9.8e-13	1.3e-09	9.2e-13	1.6e-11	1.3e-12	2.6e-13	1.9e-13	2.4e-13
7	0.1	**4.3e+01**	7.3e-13	**3.4e+00**	1.3e-12	3.5e-13	1.7e-13	2.8e-13	1.1e-13
7	0.2	**5.9e+01**	3.6e-13	8.5e-13	**2.9e-03**	3.3e-13	4.9e-14	1.8e-13	3.5e-13
7	∞	2.5e-12	0.0e+00	3.8e-13	**6.6e+00**	0.0e+00	6.7e-14	0.0e+00	0.0e+00
10	0.01	**2.6e-01**	6.7e-11	1.0e-10	**7.4e-01**	2.8e-11	1.2e-10	2.9e-11	2.5e-11
10	0.1	2.6e-10	**1.2e+01**	**1.5e+01**	4.7e-11	1.8e-10	2.3e-11	1.0e-10	3.5e-11
10	0.2	**2.4e+01**	**5.1e-03**	**7.2e+00**	**1.8e+00**	5.7e-11	4.7e-11	5.7e-11	6.5e-11
10	∞	**7.2e+01**	7.0e-11	9.5e-11	1.1e-10	0.0e+00	0.0e+00	3.9e-11	1.0e-10
14	0.01	**1.8e-01**	**1.3e-01**	**4.4e-01**	**4.7e-05**	6.3e-12	1.9e-12	1.8e-12	1.4e-12
14	0.1	**6.7e+00**	2.7e-11	2.8e-12	2.0e-11	3.0e-12	2.9e-12	1.7e-12	3.6e-12
14	0.2	1.1e-10	**3.4e+00**	3.3e-12	**8.0e+00**	2.2e-12	3.5e-12	3.3e-13	2.8e-12
14	∞	4.3e-12	**3.9e+01**	**3.8e+01**	2.2e-12	6.7e-14	2.7e-12	2.7e-13	3.0e-12

(a) Yang

rg(1e-)	ϵ	$\bar{p}$=0.2	$\bar{p}$=0.3	$\bar{p}$=0.4	$\bar{p}$=0.5	$\bar{p}$=0.6	$\bar{p}$=0.7	$\bar{p}$=0.8	$\bar{p}$=0.9
4	0.01	5.7e-12	6.2e-10	6.4e-13	**7.4e-02**	2.4e-13	**1.3e-02**	2.0e-13	1.3e-13
4	0.1	2.7e-13	4.7e-13	4.1e-13	4.1e-13	**2.0e-02**	**1.9e-02**	**1.2e-02**	**1.3e-02**
4	0.2	**3.3e+01**	**2.7e+01**	**1.9e+00**	**3.1e+01**	2.7e-13	1.8e-13	2.6e-13	**1.6e-02**
4	∞	**7.7e-02**	1.7e-13	2.0e-13	3.8e-13	2.4e-13	1.3e-13	3.8e-13	1.5e-13
7	0.01	**5.1e-02**	5.3e-06	**3.2e-05**	**1.3e-05**	3.5e-07	**5.1e-05**	**1.2e-05**	**1.6e-05**
7	0.1	**4.8e+00**	9.6e-06	**7.4e-05**	1.4e-06	4.0e-07	1.4e-06	1.9e-08	1.0e-06
7	0.2	**1.7e+01**	**2.2e-02**	6.2e-08	2.6e-06	1.5e-06	2.3e-07	5.7e-06	1.0e-08
7	∞	5.7e-06	**2.8e-05**	5.9e-10	4.4e-06	2.2e-09	2.8e-06	5.5e-10	1.1e-07
10	0.01	**1.6e+00**	**1.4e-02**	1.1e-07	1.8e-08	4.8e-08	3.6e-08	3.2e-08	1.9e-08
10	0.1	3.6e-08	4.7e-08	1.8e-08	3.0e-08	1.8e-08	5.0e-08	1.1e-08	7.3e-09
10	0.2	**1.7e-05**	**7.4e-02**	6.5e-08	9.3e-08	5.0e-08	1.7e-08	1.0e-08	1.6e-08
10	∞	**3.8e+01**	**3.0e+00**	2.3e-09	2.4e-08	1.8e-08	2.9e-08	9.0e-09	8.8e-09
14	0.01	**1.0e+00**	**2.2e-01**	5.4e-12	4.6e-12	4.5e-12	7.4e-12	2.1e-12	2.1e-12
14	0.1	**7.0e+00**	1.4e-08	5.6e-12	4.6e-12	3.9e-12	2.9e-12	7.5e-12	4.5e-12
14	0.2	**5.2e+01**	6.9e-12	**4.4e-05**	2.3e-12	3.2e-12	6.2e-10	4.7e-12	6.1e-11
14	∞	**1.3e+00**	4.4e-12	6.3e-12	1.1e-12	0.0e+00	2.3e-12	3.2e-12	2.3e-12

(b) Sioux-Falls Small

Table 4.3: PRMSE measure. Values greater than $1e-5$ are reported in bold face.

to be achievable starting from a coverage pattern with $\bar{p}$ greater than 0.5, although the black-box problem is often solved at the optimum.

In Table 4.4 a comparison in terms of PRMSE of the algorithm ISMO-CD is performed with the results reported in [59], where a bi-level programming approach developed by Patriksson et al. [29] is compared with the single-level methods proposed by Nie et al. [46] and Shen et al. [59]. In these results,

Network	Algorithm	ϵ	$\theta < 50\%$	$50\% \leq \theta \leq 75\%$	$\theta > 75\%$
Y	Patriksson et al.		18.14	0.44	0.01
	Nie et al.		59.47	35.84	33.25
	Shen et al.		63.87	5.53	6.41
	ISMO-CD	0.01	1.05	0.14	7.2e-12
	ISMO-CD	0.1	7.73	2.3e-11	1.7e-11
	ISMO-CD	0.2	11.72	0.82	1.5e-11
	ISMO-CD	∞	21.58	0.55	1.7e-11
SFS	Patriksson et al.		22.38	8.68	2.33
	Nie et al.		46.86	46.86	6.00
	Shen et al.		33.54	12.30	0.00
	ISMO-CD	0.01	0.24	7.3e-03	3.5e-06
	ISMO-CD	0.1	0.98	3.2e-03	3.1e-03
	ISMO-CD	0.2	10.92	2.58	2.0e-03
	ISMO-CD	∞	3.53	6.0e-07	1.6e-08

Table 4.4: PRMSE for three different classes of coverage.

three different levels θ of coverage are reported. For the considered algorithm ISMO-CD, we report the average results with respect to the four different values of ϵ computed considering all the levels of coverage and all the precisions of the TAP algorithm. In such a way, the comparison with the other methods is not affected by different strategies concerning the computation of the historical matrix. The most unfavorable case for ISMO-CD is indeed the case with $\epsilon = \infty$ and one may refer to it for the comparison.

Despite the simplicity of the method, the results reported in Table 4.4 show that ISMO-CD is able to obtain very accurate estimations of the true O/D demand matrix compared to the other methods, regardless the level of coverage θ and the uncertainty of the historical matrix related to the values of ϵ.

4.4.6 Numerical results of D-ISMO-CD

The evaluation of the decomposition approach D-ISMO-CD is performed with respect to the case without decomposition ISMO-CD. In this section we aim to evaluate both the effectiveness of this strategy and the impact as the size of the problem increases. For what concern the comparison with

ISMO-CD, the overall number of function evaluations of each test has been maintained fixed in order to ensure fairness. For the latter goal, since Sioux-Falls counts 528 O/D demands, several sub-networks can be defined.

In Table 4.5 six different networks have been derived from Sioux-Falls, in which SF528 corresponds to SF and the other have been built maintaining the network and subsampling the O/D matrix. For each problem size, several space decomposition patterns have been tested with the same break-down rule used for the subsampling of the O/D matrix. In such a way, the test bed can be easily illustrated in Table 4.5 and in the following. Furthermore, for each test and for each block count, a number of Gauss-Seidel iterations and maximum evaluations per run have been set in order to ensure that the overall number of evaluations will be the same for each problem size. Finally, the case with one block corresponds to ISMO-CD and no external Gauss-Seidel iteration is performed in this test.

For what concern the employment of ISMO-CD inside of D-ISMO-CD, the following settings have been used:

$$(\epsilon = 0.1, \bar{p} = 0.5, rg = 10^{-12})$$

Each result of D-ISMO-CD is the average of ten runs and the mean of objective function $F(\cdot)$ and PRMSE has been collected in Tables 4.6 and 4.7, respectively.

For what concern the performances from the black-box optimization point of view, the results show the effectiveness of the approach. We can observe that the cases with a small number of variables in each block (from 2 to at most 6) are the best ones. The gap between D-ISMO-CD and the 1-block optimization is more evident as the size of the network increases. Although the number of evaluations is the same for all the tests, the approach with decomposition may take advantage of several subsequent runs of the black-box optimization on different and separate parts of the variables space. Basing on these considerations, we have performed further tests with one block increasing the outer number of Gauss-Seidel iterations while reducing the number of black-box evaluations of each run in order to maintain the fixed number of total evaluations. The aim of this test is to restart the coordinate descent method from the current iteration several times, restoring the initial mesh size.

The results of this test are not reported here since significant differences with respect to the case with one single run have not be observed. Thus, we can conclude that the main benefit of the decomposition consists in the

Networks		# blocks						
		1	11	33	66	132	264	528
SF528	# iterations	1	32	24	16	12	8	8
	# run evals	152064	432	192	144	96	72	36
	# total evals	152064	152064	152064	152064	152064	152064	152064
SF264	# iterations	1	32	16	12	8	8	
	# run evals	76032	216	144	96	72	36	
	# total evals	76032	76032	76032	76032	76032	76032	
SF132	# iterations	1	24	12	8	8		
	# run evals	38016	144	96	72	36		
	# total evals	38016	38016	38016	38016	38016		
SF66	# iterations	1	12	8	8			
	# run evals	19008	144	72	36			
	# total evals	19008	19008	19008	19008			
SF33	# iterations	1	8	8				
	# run evals	9504	108	36				
	# total evals	9504	9504	9504				
SF11	# iterations	1	8					
	# run evals	3168	36					
	# total evals	3168	3168					

Table 4.5: Test bed for the evaluation of the decomposition approach

Networks	# blocks						
	1	11	33	66	132	264	528
SF528	2330	127	16.5	3.49	3.36	**2.52**	5.01
SF264	4830	89.9	26.2	26.7	**16.4**	36	
SF132	1520	60	**5.24**	7.59	8.51		
SF66	318	**3.83**	9.75	10.5			
SF33	257	2270	**3.62**				
SF11	0.0699	**0.0396**					

Table 4.6: Final objective function values. Best values are reported in bold face.

Networks	# blocks						
	1	11	33	66	132	264	528
SF528	125.0	**75.2**	86.3	90.9	97.8	96.3	82.4
SF264	256.0	88.2	82.0	**68.6**	87.7	74.4	
SF132	127.0	75.1	55.6	67.5	**47.6**		
SF66	65.4	27.6	**26.1**	31.1			
SF33	19.1	18.4	**16.2**				
SF11	**7.84**	8.94					

Table 4.7: PRMSE measure. Best values are reported in bold face.

alternation of several optimizations on small sets of variables. Anyway, a rigorous analysis of the reasons underneath the performances of D-ISMO-CD have not be performed yet and we are currently trying to confirm such results on a general benchmark with several well-known test problems in the field of derivative-free optimization.

The estimation capability of the approach is improved as well, as it can be observed in Table 4.7. However, differently from the case of the test networks Yans and Sioux-Falls Small, the Percentage Root Mean Square Error (PRMSE) is considerably high and it generally increases as the number of O/D pairs increases. As an example, on SF528 the test with the best PRMSE is able to estimate the O/D matrix demand with an average error of 75% on each demand variable.

The parameter $\bar{p} = 0.5$ may lead to these poor results from the estimation point of view, thus we performed another test, here not reported, in which the link coverage has been extended to 80%, which is unlikely in real scenarios. However, although the estimation capability improves, the best PRMSE on SF528 is nearly equal to 50%.

From these test on Sioux-Falls (528 O/D pairs) we can observe that, while the optimization problem can be solved even for real networks, the estimation task is still hard in case of a number of O/D pairs greater than a few dozens.

In conclusion, since the networks SF11, SF33, SF66, SF132, SF264 have been built subsampling the original O/D demand of network Sioux-Falls, the results and the complexity of the estimation problem may be affected by the sampling approach. Furthermore, considering only a part of the O/D pairs means obtaining a new *user equilibrium* solution which is fundamentally different from the original one. Thus, we have followed another approach

in which the O/D matrix is not sampled and the variables not considered in the optimization are maintained fixed to the true value. For example, in case of SF11, the estimation involves 11 variables, while the remaining 517 are fixed. In such a way, no fictitious element might be inserted from the subsampling process.

Anyway, the results obtained generally confirm what have been observed for both the estimation and the optimization capabilities of D-ISMO-CD and ISMO-CD.

Chapter 5

Zone Planning Problem

Starting from a robust O/D demand estimation framework, transportation planners can deal with several modelling scenarios. One of these could be the design of new transportation systems. Since a new public transportation solution has to satisfy both customers and companies needs, the knowledge of how many users actually travel from an origin to a destination has a significant relevance when the effect of a variation has to be evaluated.

For a transportation planning task like the one described in this chapter, i.e., a zonal-based tariff system, one may take advantage of the development of a demand estimation framework in order to build a public transportation O/D demand matrix, rather than, for example, relying on outdated surveys.

This chapter is structured as follows. In Section 5.1, an introduction of the problem is reported. In Section 5.2, the zone planning problem is introduced and described while in Section 5.3 a brief analysis of the literature on zone planning methods is reported. In Section 5.4, the proposed algorithm is described, along with an overview on metaheuristics. Finally, computational results are reported and discussed in Section 5.5.

5.1 Introduction

The planners of public transport systems have shown a rising interest in promoting the integration of different means of transport and companies in order to improve the attractiveness of their offers (see, e.g., [27, 57]). Nowadays the integration is commonly achieved simplifying and unifying the ticket and the tariff system. While traditional tariff plans were based on

distance traveled and on special marketing strategies, zone tariff plans are based on a partition of the public transport network into a finite number of zones and the price of a ticket just depends on the number of zones which have to be traversed during a trip from origin to destination. An increasing number of municipalities in Europe switched to this system or are planning such a change of pricing policy (see, for example, the fare systems in many German cities like Köln, Frankfurt, München, or the regional fare system in the Saarland's area studied in [27]). We were faced ourselves with this interesting problem for a study regarding the design of tariff zones for the Tuscany region of Italy. There exist also alternative fare systems, even within those based on zones, in which the fare depends explicitly on the origin and destination zones. Although a methodology like the one proposed here might in principle be extended to this case also, we did not pursue this here.

In [21] a set of methods is presented, which can be used to approximately solve the zone planning problem. This problem is characterized by two levels: in the first level, assuming a partition of the network into zones has been already performed, the problem to be solved is that of deciding the cost of the ticket required to travel between any two zones. This decision should take into account the conflicting objectives of transport companies, who do not want to lose income, and of users, who are not willing to pay more than they used to pay with the original system. It has already been shown [57], and here we generalize the result, that this problem is easily solved. At the second level, given the rule to generate "optimal" tariffs, the problem becomes that of designing the zones. Here we propose a local-search based heuristic which produces very good results in reasonable time, when compared with the few existing algorithms in the literature.

5.2 Problem definition

Let $G(N, A)$ be the graph of a Public Transportation Network (PTN), where the node set N represent single bus stops, or demand centroids or even whole municipalities. An arc exists between any pair of nodes directly connected by a public transport line. By this we mean that an arc connects two nodes $i, j \in N$ if and only if there is at least one way by which, through public transport, it is possible to travel from i to j or viceversa without touching any other node in N. If N are bus stops, arcs connect stops which are adjacent along at least a bus line. If N is composed of regions (e.g., zones

with the same zip code), arcs connect two zones if a bus connection exists between the two zones which does not need to touch any other region.

Given $L \in \mathbb{N}$, any partition of nodes in G, $Z = \{Z_1, Z_2, \ldots, Z_L\}$, such that

$$\begin{aligned} Z_1, Z_2, \ldots, Z_L &\subseteq N \\ Z_i \bigcap Z_j &= \emptyset \qquad\qquad \forall\ i \neq j \\ \bigcup_{i=1}^{L} Z_i &= N, \end{aligned}$$

might be called a zone partition; usually, however, the subgraph of G induced by each zone Z_i,

$$(\ Z_i, \{(u,v) \in A : u \in Z_i, v \in Z_i\}\ ),$$

is assumed to be connected, i.e. we require that, once a zone has been defined, it is possible to travel between any two nodes in the same zone without having to cross a zone border.

Given any two nodes $u, v \in N$, let n_{uv} the minimum number of zones borders to be crossed along a path in G from u to v. Given a zone partition, the zone planning problem consists in deciding the price $c(n)$ to be paid by users whose shortest path from origin to destination crosses $n \geq 0$ zones borders; of course the case $n = 0$ corresponds to trips which are included in a single zone. By "shortest path" in this context we mean the path which requires the minimum number of zone border crossing, as the total tariff to be paid increases with the number of zones. Any user traveling from node u to node v has to pay a ticket whose price is $c(n_{uv})$. We assume that customers travel along shortest paths from origin to destination, even if it might happen, however, that other factors, different from shortest paths, might influence customer choices, like traveling time, ticket price or the number of changes of means of transportation along a path. Here we assume that users travel along a path with minimum number of zone crossing, as this also corresponds to minimum cost paid per travel.

Let $D \in \mathbb{N}^{|N| \times |N|}$ be the Origin-Destination (O/D) matrix with estimates of transport demand D_{uv} from origin u to destination v and let P_{uv} be a reference price paid by users before the introduction of the new tariff system. This reference price might be a weighted average of prices paid when using different companies between the two nodes, or even a weighted average

between single ride and special tariff tickets (like student fares, monthly fares, subscription or season tickets). A new fair tariff plan is expected to minimize the deviations between the new tariff and the reference prices. As this is a multi-objective problem, usually it is transformed into a single aggregate one. Let, for every $(u, v) \in N \times N$, W_{uv} be non negative weights. Typical examples are

$$b_\infty = \min_c \max_{u,v \in N} W_{uv} |c(n_{uv}) - P_{uv}| \qquad \ell_\infty \text{ objective} \qquad (5.1)$$

or

$$b_1 = \min_c \sum_{u,v \in N} W_{uv} |c(n_{uv}) - P_{uv}| \qquad \ell_1 \text{ objective} \qquad (5.2)$$

In [21], the ℓ_∞ objective function is considered, as this is perceived as more fair with respect to users. In such objective function, weights W_{uv} can be chosen in many different ways:

- $W^D_{uv} = D_{uv}$: new tariffs are defined such that maximum deviation in total income for transport companies is minimum as possible;
- $W^U_{uv} = 1$: new tariffs are defined such that maximum deviation among old and new fares is minimum as possible;
- $W^R_{uv} = 1/P_{uv}$: apparently ignored in literature, relative price variation is considered as opposed to the absolute variation.

It is very important to notice that the model we are proposing is fairly general and that, by just changing the value of the weights W, different objectives can be optimized, ranging from a user-centric perspective in which tariffs change as little as possible for the users, to a company-based one in which total company income are kept as stable as possible. Of course by changing these weights, different compromise solutions can be found as it is typical of multi-objective models. What is important to remark here is that, given the weights W, the analysis we perform as well as the algorithms we develop and propose are unchanged.

Whichever the weight chosen, it is easy to show that the optimal fare can be easily found. Given a zone partition Z, let $M_n(Z)$ be the set of pairs of nodes whose distance (minimum number of zones to be crossed) is n.

Theorem 5.1. *Let $Z = \{Z_1, Z_2, \ldots, Z_L\}$ a given zone partition and let*

$$b_\infty(n) = \min_c \max_{(u,v) \in M_n(Z)} W_{uv} \, |c(n) - P_{uv}|$$

the ℓ_∞ – objective as a function of the number of zone to be crossed n. Define

$$\omega((u,v),(w,x)) = \frac{W_{uv}W_{wx}}{W_{uv}+W_{wx}}(P_{uv}-P_{wx}), \tag{5.3}$$

then

$$b_\infty = \max_n b_\infty(n)$$

and optimal fares are

$$c^\star(n) = \frac{W_{u^*v^*}P_{u^*v^*} + W_{w^*x^*}P_{w^*x^*}}{W_{u^*v^*}+W_{w^*x^*}}$$

where

$$((u^*,v^*),(w^*,x^*)) \in \arg \max_{(u,v),(w,x)\in M_n(Z)} \omega((u,v),(w,x)) \tag{5.4}$$

The proof, which is quite elementary, is omitted (it can be derived generalizing a similar one published in [27, 57]). A quite similar result exists also for the ℓ_1 objective function.

As it can be seen from the above theorem, finding optimal fares is an easy problem, once criteria and weights have been defined and the zones have been determined. It is worth observing, however, that the problem of assigning fares, although easy, has a complexity which might be non negligible when the graph G is large: a straightforward implementation requires $O(|N|^4)$ operations, which includes shortest paths computation, $M_n(Z)$ sets definition and optimal fares computation.

Summarizing previous observations, the formulation of the problem of solving the zone planning problem is stated in Definition 2.

Definition 2 (Zone Planning Problem)**.** Let $G(N,A)$ be a PTN graph, $D \in \mathbb{N}^{|N|\times|N|}$ be the O/D matrix with demand D_{uv}, $\forall\ u,v \in N$, P_{uv} and W_{uv} be respectively reference prices and non negative weights, $\forall\ u,v \in N$. Given $L \in \mathbb{N}$ and assuming ℓ_∞ as the objective function, the zone planning problem can be defined as finding a zone partition

$$Z^* = \{Z_1^*, Z_2^*, \ldots, Z_L^*\}$$

such that

$$Z^* \in \arg\min_Z \quad \max_n \max_{(u,v),(w,x)\in M_n(Z)} \frac{W_{uv}W_{wx}}{W_{uv}+W_{wx}}(P_{uv}-P_{wx})$$

and the subgraph of G induced by zone Z_i^*, i.e.

$$(Z_i^*, \{(u,v) \in A : u \in Z_i^*, v \in Z_i^*\}),$$

is connected, $\forall\ i = 1, \ldots, L$.

Note that in a fair solution n can vary from zero, i.e. the set of pairs of nodes within the same zone, to at most $L-1$, when traveling from an origin to a destination requires crossing all planned zones.

As it is proven in [27, 57], the Zone Planning Problem is NP-hard. Once a partition of G is fixed, a zone graph $G_Z = \langle Z, A_Z \rangle$ can be defined, in which the nodes are the zones and $A_Z \subseteq Z \times Z$ are arcs connecting two adjacent zones, i.e. pairs of zones for which there exists a direct transport connection.

The focus of the work is on the ℓ_∞ objective function and its properties are exploited to obtain better search speed and quality.

5.3 Related work

It is quite surprising that a planning problem of this kind, which involves medium to long-term planning horizon, with relevant investment decisions, constraints, conflicting objectives, has received such a little attention in the operations research community.

The main contributions on zone planning are reported in [5, 27, 57]. Reference [27, 57] have already been cited and they are the most relevant ones in the literature; in [5] the problem is mainly analyzed from the point of view of complexity and several theoretical results are proven. In [2] the need of an integrated tariff system is due to pollution and traffic congestion issues, so the aim is to make the tariff system more attractive for users. In addition, different approaches exist in literature. In [31] a *price-oriented* tariff system is proposed for a simplified and easy to understand *distance-price* scheme, in which prices are a piece-wise constant function of the distance to be traveled with fixed differences between consecutive prices.

Three heuristic methods have been proposed in [57] for the problem of fare zone design, with some numerical evidence reported. Here we discuss the ideas behind these methods.

Sequential Agglomerative Hierarchical Nonoverlapping (SAHN) clustering: the algorithm starts by assigning each node in N to a different zone;

then, until a prefixed number L of zones has been obtained, using a dissimilarity criterion, two zones are clustered into a single one. Initially, pairs of zones Z_i and Z_j are compared on the basis of the reference price P_{ij}, i.e., $d(Z_i, Z_j) = P_{ij}$. Then, zones are merged following one of the following criteria:

- *single linkage*: Z_x and Z_y separated by the shortest distance are merged into a new zone Z_k and dissimilarity measures are updated as follows

$$d(Z_h, Z_k) = \min\{d(Z_h, Z_x), d(Z_h, Z_y)\}, \quad \forall\, h.$$

- *complete linkage*: Z_x and Z_y separated by the farthest distance are merged into a new zone Z_k and dissimilarity measures with each other zone Z_h are updated as follows

$$d(Z_h, Z_k) = \max\{d(Z_h, Z_x), d(Z_h, Z_y)\}, \quad \forall\, h.$$

Other possibly criteria exist.

Greedy heuristic: the method starts, similarly with the previous one, assigning one zone to each node and proceeds by sequentially merging two zones into a single one guided by the criterion of minimizing the resulting ℓ_∞ objective function.

Spanning tree heuristic: unlike the other two, the algorithm starts with a single zone containing all nodes. On the graph G equipped with the reference prices P, a maximum weight *spanning tree* is built. Then, the most costly arcs of the spanning tree are removed until L connected components remain.

In [21] is shown, and here is reported, how to build a local search heuristic which outperforms these methods both from the point of view of the quality of the resulting subdivision and for the CPU time required. It might seem surprising that CPU is a criterion to evaluate the goodness of a zone design algorithm, as this is a long-term planning problem, which might be solved, possibly, once every few years. However, standard implementations of elementary heuristics might require, for graphs of moderate to large dimension, even several hours of CPU time; an exact model, based on mathematical programming, would require an enormous amount of CPU time and in general

it may not be able to close the duality gap to a sufficiently small value, as we had the opportunity to see in a few trials with tiny dimensional instances. In fact, the unique published approaches in the literature related to this subject are devoted to heuristics. As the fare planning problem is in practice a multi-objective one in which the contrasting objectives of both the users and the transport companies have to be taken into account, having a relatively efficient algorithm is fundamental in order to give transport managers a tool which can be used to simulate different scenarios and to experiment with different weights before arriving to the final decision.

5.4 Algorithm description

The proposed method arises from classical local search based metaheuristics. Using ℓ_∞ as the objective function, the algorithm consists in an Iterated Local Search (ILS) [38], in which each local search is performed by a Tabu Search (TS), as proposed in [26] to overcome local optima during search.

Among the huge literature on tabu search, we recommend [24,26], where the main features of TS are described: allowing nonimproving moves and avoiding cycling back to previously visited solutions by the use of memories, called tabu lists, that record the recent history of the search.

As TS is an extension of classical local search methods, the search space and the neighborhood structure must be defined. For zone planning problems, the search space is made of all zone partitions of network graph G such that corresponding zone graphs Z_g are connected and each zone is a connected sub-graph of G. The simplest possible neighborhood structure is considered, i.e. the one associated to moves corresponding to the displacement of a single node $u \in N$ from its current zone to a destination node, which might be an adjacent one, or a new one, thus allowing for the creation of a new zone. Moves that lead to the emptying of a source zone are considered too, whose effect is to remove zones from the partition. In such a context, a move is considered tabu as well as its inverse; in the special case in which a move empties a zone, the inverse move is considered as that which moves that node from its current zone and creates a new one consisting of that node only. With respect to the problem definition given in Section 5.2, where the number of zones L is fixed, this neighborhood structure allows for the increase or decrease of L. In such a way it is possible to overcome the limitation of a fixed and pre-defined value for L.

Once the neighborhood structure is defined, several neighborhoods can be built. Here we present a set of possible choices.

Complete 1–neighborhood : composed of all possible single node moves;

Unaltered Zone Graph neighborhood (UZG) : if we associate to each arc of the zone graph G_Z (corresponding to a zone partition of G) a cost equal to one, it can be easily shown (see. e.g., [27,57]) that shortest paths among all pairs of nodes in G, in terms of number of zone edges to be crossed, are equivalently calculated using G_z. Thus, if a move does not change G_z, shortest paths do not need to be recomputed and a faster computation of ℓ_∞ can be obtained.

Greedy neighborhood (GRDY) : subset of the 1–neighborhood composed of moves which affect nodes involved in the definition of the ℓ_∞ value. As the objective function is computed as

$$b_\infty = \max_n \max_{(u,v),(w,x)\in M_n(Z)} \omega((u,v),(w,x)) \quad = \omega((u^*,v^*),(w^*,x^*)),$$

where $\omega((u,v),(w,x))$ is defined in (5.3), the aim is to consider a set of moves that involves only nodes u^*, v^*, w^*, x^*, in a greedy attempt to separate pairs (u^*, v^*) and (w^*, x^*).

The complete 1–neighborhood, is computationally expensive when the size of the network increases. Instead, the GRDY neighborhood contains a few moves, mainly dependent on the connection degree of nodes u^*, v^*, w^*, x^*. Furthermore, GRDY leads in practice to good solutions in few iterations and is preferred over the other neighborhoods. Although the UZG neighborhood is bigger than GRDY and less effective, its properties allow for a fast computation of all moves and can be used for a refinement of the current solution.

Thus, based on previous considerations, our algorithm developed for the zone planning problem, *Zone Planning Iterated Local Search* (*ZP-ILS*), is described below in Algorithm 11.

Given a random solution with an estimate of the desired number of zones L, in which each node is assigned to a zone using a uniform distribution, the main cycle corresponds to the ILS scheme with the definition of a termination criterion. Its body performs a TS with neighborhood GRDY followed by a perturbation of the current solution, which possibly allows to escape from a local minimum.

Algorithm 11: Zone Planning Iterated Local Search *ZP-ILS*

Data: a feasible solution s_0 consisting of L zones

$s^* \leftarrow s_0$;
$k \leftarrow 0$;
while *a termination criterion is not met* **do**
  $\hat{s}_k \leftarrow$ TS <GRDY>(s_k);
  $s^* \leftarrow AcceptanceCriterion(s^*, \hat{s}_k)$;
  $s_{k+1} \leftarrow Perturbation(s^*)$;
  $k \leftarrow k + 1$;
end
$\hat{s}^* \leftarrow$ TS<UZG>(s^*);
$s^* \leftarrow AcceptanceCriterion(s^*, \hat{s}^*)$;

In the end, a tabu search with neighborhood UZG is performed, in order to refine the final solution. It is worth noticing that each local search is followed by the application of an acceptance criterion of the solution found with respect to the best one. We perform the simplest criterion, i.e., TS solution is accepted, with a fixed tolerance, if it is better than the current best one.

5.5 Computational results

Firstly, we describe the datasets used for the experiments. Then some implementation details are reported, as well as the adopted experimental environment. Finally, results are presented and discussed.

Datasets

In order to solve the zone planning problem several information are needed, such as the PTN graph, reference prices and the estimated transport demands between pairs of nodes.

Two real datasets were used for experiments. The first was extracted from the Metropolitan Area of Florence where the 73 nodes are municipalities. Reference prices were collected from kilometric tables of the local transportation companies. The O/D demands were available from a survey. The other dataset was extracted from the Washington Metropolitan Network

[4]. The network is made of 91 stations and 118 connections and the reference prices were collected on the website. The O/D matrix was built starting from a survey of rail station-to-station passenger counts in May 2012, available at the metro's planning blog [8]. Several O/D matrix are available: we chose the survey made in week days at the AM peak.

However, as we discuss later, the case of the Washington Metropolitan Network is critical for the considered zone planning model since, except for a few central stations, which are well connected, the peripherical stations can be reached only through dedicated metro lines. Thus, it can be expected that a large number of small zones (even made of a single station) can be arranged around a big central zone.

Since these two datasets are of small size and obtaining a comprehensive set of real data of arbitrary size was not possible, we defined a process to generate our own synthetic datasets, under suitable assumptions. First, we extracted graphs from existing road networks, then we generated population π_i of each node i according to an exponential distribution

$$\pi_i \sim Exp(\lambda).$$

with λ equal to the reciprocal of the desired population mean. Reference prices P_{ij} for trips between node pairs were assigned using a set of kilometric tables. Finally, given π_i, π_j population of nodes i, j, reference price P_{ij} and positive constants a, b, we estimated the O/D demand D_{ij} with the following formula

$$D_{ij} = a\frac{\pi_i \pi_j}{P_{ij}^b},$$

known as *gravity form* [49]; this model has been used in several trip distribution models in order to derive origin/destination flows proportional to population sizes and inversely correlated with transportation cost; in this context, the model was used in order to derive a reasonable O/D matrix based on a simple, widely accepted, generation tool.

Graphs were extracted from Manhattan (375 nodes) and Brooklin (636 nodes) using OpenStreetmap data [48]. For each graph we generated a dataset with 10% and 50% of nonzero entries in O/D matrix D, which represent travel flows, for a total of four synthetic datasets. All the datasets are summarized in Table 5.1.

Dataset	# nodes	# arcs	# O/D pairs
Florence	73	186	5329
Washington	91	118	8281
Manhattan10	375	543	14062
Manhattan50	375	543	70312
Brooklin10	636	1000	40449
Brooklin50	636	1000	202248

Table 5.1: Details of the datasets used for the computational experiments.

Implementation details

The algorithm was implemented in C++ [14] and compiled using the GNU Compiler Collection [60] version 4.4.3 using the `-O2` optimization flags, METSlib [39] framework for the tabu search logic, and the OpenMP [47] extensions for simple code parallelization. All the instances were solved on a workstation equipped with an Intel Core(TM) i7-870 CPU and 8 GB of main memory.

Termination criterion of ILS in Algorithm 11 is based on a maximum number of iterations (set to 25), while both the tabu searches TS use a termination criterion based on the maximum number of non improving iterations (set to 25). The tabu list tenure is of 10 elements and an accepted move is made tabu for the next iterations along with its opposite. Tabu moves can be accepted if they satisfy a so-called *aspiration criterion*: in this case, an improvement of the current best solution has the effect to accept the move, even if it has been marked as tabu.

Results

The described algorithm *ZP-ILS* were compared with the heuristics *greedy*, *clustering* (SAHN) and *spanning tree* reported in [27, 57] and described in Section 5.3, using the ℓ_∞ objective function and weights W^D_{uv}, W^U_{uv}, W^R_{uv} as defined in Section 5.2. Results are shown in Tables 5.2-5.4 where best values are reported in bold face.

Results show that *ZP-ILS* was able to produce the best solution most of the times, on real and synthetic datasets. In those cases in which *ZP-ILS* was not able to produce the best solution, the returned solution is quite close

	greedy		*clustering*		*spanning tree*		*ZP-ILS*	
	ℓ_∞	time	ℓ_∞	time	ℓ_∞	time	ℓ_∞	time
Florence	478.9	5s	342.2	<1s	478.9	<1s	**266.1**	<1s
Washington	664.5	12s	1389.1	<1s	1047.4	<1s	**329.0**	6s
Manhattan10	108.4	26m 54s	108.9	<1s	118.1	<1s	**78.1**	17m 57s
Manhattan50	235.3	3h 53m	214.0	5s	242.2	4s	**180.0**	7h 39m
Brooklin10	101.9	5h 0m	113.0	1s	121.8	1s	**69.6**	5h 22m
Brooklin50	261.3	48h 48m	239.9	20s	209.2	19s	**188.7**	15h 29m

Table 5.2: Results on real and synthetic datasets with W^D_{uv}.

	greedy		*clustering*		*spanning tree*		*ZP-ILS*	
	ℓ_∞	time	ℓ_∞	time	ℓ_∞	time	ℓ_∞	time
Florence	1.05	3s	1.05	<1s	1.3	<1s	**0.975**	1s
Washington	1.875	2s	1.875	<1s	1.875	<1s	**0.796**	7s
Manhattan10	2.550	7m 28s	2.550	<1s	2.225	<1s	**2.000**	12m 24s
Manhattan50	**5.675**	19m 27s	7.200	<1s	7.200	<1s	6.150	23m 42s
Brooklin10	2.750	1h 12m	2.925	<1s	2.750	<1s	**2.550**	22m 2s
Brooklin50	7.200	2h 57m	6.350	<1s	6.500	<1s	**5.600**	47m 30s

Table 5.3: Results on real and synthetic datasets with W^U_{uv}.

	greedy		*clustering*		*spanning tree*		*ZP-ILS*	
	ℓ_∞	time	ℓ_∞	time	ℓ_∞	time	ℓ_∞	time
Florence	0.512	1s	0.5	<1s	0.538	<1s	**0.458**	2s
Washington	0.465	2s	0.465	<1s	0.465	<1s	**0.229**	5s
Manhattan10	0.689	7m 51s	0.613	<1s	**0.578**	<1s	0.631	18m 31s
Manhattan50	0.832	19m 16s	0.643	<1s	0.754	<1s	**0.613**	27m 5s
Brooklin10	0.689	1h 14m	0.659	<1s	0.613	<1s	**0.564**	36m 3s
Brooklin50	0.762	3h 6m	0.753	<1s	0.745	<1s	**0.600**	1h 1m

Table 5.4: Results on real and synthetic datasets with W^R_{uv}.

to that of the best performing method; on the contrary, in many cases *ZP-ILS* is capable of generating a solution whose quality is significantly better than that produced by the other methods we tested. Moreover, when the number of nodes increases, the execution time of the heuristic *greedy*, which is considered the best one in practical cases, as it is shown in [27, 57], grows more rapidly than *ZP-ILS*, as we can observe mainly in Table 5.2.

On the real dataset of Washington Metropolitan the proposed algorithm produces solutions which are considerable better than the ones obtained by the three heuristics. It's worth observing that the heuristics produce solutions of the same cost, respectively for the weights W^U_{uv} and W^R_{uv}. From tests, it turns out that these costs are the same for every possible number of planned zones, i.e., $\forall \;\; L \in [1, |N|]$. Thus, the heuristics show a lack of robustness in this case compared to the algorithm *ZP-ILS*, which, on the contrary, produces solutions of good quality. Indeed, for the W^U_{uv} weight, a maximum variation of 0.796 \$ is obtained on a single ticket for each O/D pair, while, for the W^R_{uv} weight, the maximum percentual variation is of 22.9%, which is significant compared to the variations obtained on the synthetic datasets. However, the number of planned zones is too large compared to the number of stations. As an example, the solution with weight W^U_{uv} is made of 39 zones on a network of 91 stations. As expected, the biggest zone includes all the central stations and smaller zones are arranged along the different metro lines.

The proposed model is more suitable for a dataset like the Metropolitan Area of Florence, where neighboring municipalities are connected properly by the transportation companies. Thus, it's easier to define a zone made of nodes geographically close which can be reached each other without exiting the zone itself. The efficacy of *ZP-ILS* compared to the three heuristics is less evident on this dataset, however it's able to achieve the best result for all weights. In Figure 5.1 we report a solution obtained on this dataset with weight W^U_{uv} [1].

For this solution, the price table is reported in Table 5.5 for a number of zones to be crossed varying from 0 (the trip starts and ends in the same zone) to 6.

	0	1	2	3	4	5	6
Ticket cost (€)	1.8	2.12	2.72	3.17	4.02	5.15	6.13

Table 5.5: Price table for the solution reported in Figure 5.1.

Although the analysis of the real impact on transportation companies

[1] It's worth observing that Figure 5.1 has the purpose to show qualitatively a division in zones of the municipalities. The reported tessellation, however, does not respect at all the contiguity of zones, which is represented by the arcs connecting the zones.

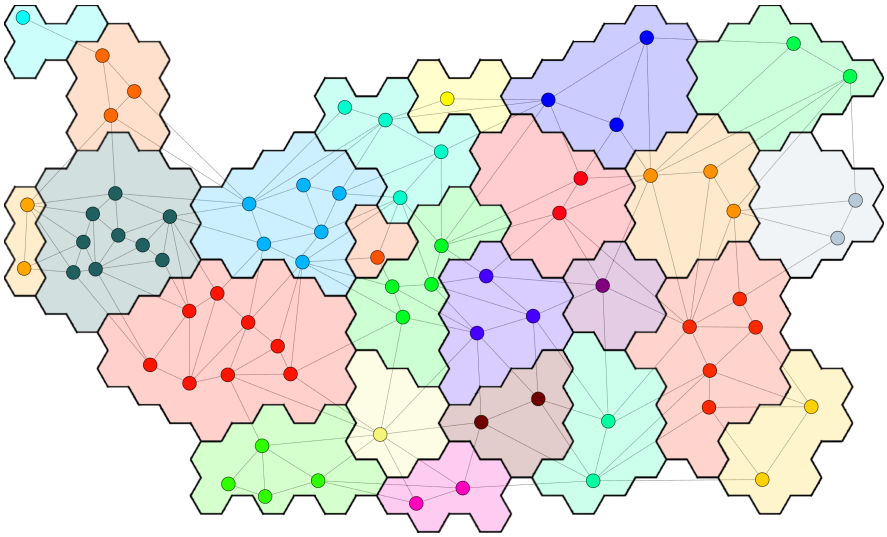


Figure 5.1: Zone partition solution on the Metropolitan Area of Florence with W^U_{uv}.

cannot be easily done observing the results in the case of weight W^D_{uv}, for the results concerning the weights W^U_{uv} and W^R_{uv} we observe that, mostly on the synthetic datasets, the deviation from the reference prices is significant while the use of a relative price variation weight like W^R_{uv} seems to be more fair, leading to a variation of approximately 50% of the reference prices over all synthetic datasets.

Furthermore, a significant computational gap can be observed between using the weight W^D_{uv} instead of the other two weights W^U_{uv} and W^R_{uv}. The reason is both mathematical and computational. It can be shown that the computation of (5.4) can be speeded up with some mathematical trick regardless the choice of the weight. Moreover, if the weight chosen is W^U_{uv} or W^R_{uv}, the computation is even faster.

Our numerical experiments show that there are several solutions and improvements to be introduced in order to exploit better the properties of ℓ_∞. Such objective function is indeed made of large connected areas of equal cost, thus a combination of ℓ_∞ with a second objective function could be exploited in order to follow a secondary goal, without sacrificing the quality of the solution for what concerns the primary objective. For example, the

primary goal could be, for customers, the minimization of the maximum variation among old and new fares, while the secondary goal could take into account the total income for transport companies in order to minimize their losses.

Finally, we observe that the tabu search's *greedy* neighborhood GRDY may contain a small number of moves, leading to a premature termination of the search. One way to add new *greedy* moves could be defining moves which consider nodes that belong to a path between (u^*, v^*) or (w^*, x^*) in order to affect the number of zone crosses needed from origin to destination.

Chapter 6

Conclusions

In this work we have defined new algorithms for transportation planning problems, with the main focus on network equilibrium problems.

The main contribution concerns the development of a new path-based method for the Traffic Assignment Problem (TAP) with inelastic demand. Its first version, Inexact Sequential Minimal Optimization (ISMO), has been introduced in [12] and described in Chapter 2, where a *Path Equilibration algorithm* (PEA) has been suitably combined with a novel column generation strategy, where the shortest paths are computed once every L iterations for each Origin/Destination (O/D) pair. With the employment of a Quadratic Line Search (QLS), where the initial step is based on first-order information, convergence properties have been demonstrated.

The results reported in 2.8 show the effectiveness of such column generation, with a significant improvement of the performances with respect to the case where shortest paths are computed at each iteration. However, the method exhibits a poor convergence capability on large networks compared to state-of-the-art path-based or bush-based methods. Moreover, numerical issues have been observed which led to an oscillatory behavior in later stages of the optimization.

Such numerical problems have been solved in a new version of ISMO developed in [22] where the initial step of QLS has been defined as the optimum of the quadratic approximation of the objective function along the chosen direction, as in Dafermos [11]. The algorithm, called ISMO Adaptive Column Generation (ISMO-ACG), introduces also an advanced adaptive column generation, where the frequency of the shortest paths computation is variable

and it is adaptively updated with respect to a measure, defined at O/D pair level, which reflects the obtained decrease of the objective function.

Moreover, this column generation strategy has been considered as a general decomposition path-based framework in which different equilibration methods can be embedded. Besides the proposed PEA with QLS and the Dafermos step, the *Gradient Projection* algorithm GP-ACG, the *Rosen Gradient Projection* method RGP-ACG and an improved version of GP-ACG, called *Incremental Gradient Projection* (IGP-ACG), have been embedded.

The results of ISMO-ACG show excellent performances from both the efficiency and the convergence capability points of view. High solution precisions have been achieved on all the tested networks within competitive computational times. Furthermore, the *Path-Equilibration* algorithm ISMO-ACG outperforms all the other path-based method with the column generation strategy being fixed.

Removing the QLS procedure, two nonmonotone versions of ISMO-ACG have been defined, ISMO-NM and ISMO-NM-A, where the latter performs an alternation of the Dafermos step and its double through the iterations. While ISMO-NM is comparable with ISMO-ACG although slightly faster, ISMO-NM-A outperforms significantly ISMO-ACG on the largest networks. Moreover, it is able to further exploit the sparseness of the equilibrium solution in terms of path flows, thus leading to extremely competitive computational performances.

Finally, for what concern the inelastic TAP, the new method IGP-ACG results to be more stable than GP-ACG and it is able to obtain more accurate equilibrium solutions.

The algorithm ISMO-ACG developed for the inelastic case has been adapted to the elastic case in Chapter 3 thanks to an equivalence result between TAP with elastic and inelastic demand. ISMO-ACG is again the more accurate and fast algorithm among the tested path-based methods and is able to reach very high solution precisions compared to the ones existing in the literature. While ISMO-NM-A has been proved to be very efficient in the inelastic case, in the elastic case it shows convergence difficulties. At the contrary, ISMO-NM, which is essentially ISMO-ACG without line search, is able to obtain impressive computational performances.

In Chapter 4, the algorithm ISMO-ACG has been used as a black-box TAP function for the estimation of the O/D demand matrix. A derivative-free black-box formulation has been defined in which the O/D matrix is

estimated while reducing the distance between real observed flows and flows obtained by the equilibration process on a sample of network links. A simple coordinate descent algorithm has been compared with some existing methods in the literature on benchmark networks and it shows good optimization and estimation capabilities. Since the benchmark networks count only a small number of O/D pairs to be estimated, a test on a real-size network with more than 500 O/D pairs has been carried out. In order to deal with such a large problem, a decomposition-based coordinate descent algorithm has been defined which outperforms significantly the basic version from the optimization point of view. However, the estimation capability of both the versions turns out to be rather poor, thus confirming that the large-scale O/D matrix estimation problem is still hard to solve.

Finally, for what concerns the transportation planning problem described in Chapter 5, the algorithm developed for the Zone Planning Problem improves significantly over the heuristics reported in the literature. The approach based on local search leads to a better exploration of the search space and it allows for an increase or decrease of the number of zones at each iteration, whereas the existing heuristics never revise the choose of the initial number of zones.

Appendix A

Quadratic Line Search

The properties of an Armijo-type line search do not guarantee, without further assumptions on the search direction d^k, that the distance between successive points tends to zero, which is a usual requirement of decomposition methods. Such a property can be satisfied by the line search algorithm described below and based on the acceptance condition

$$f(x^k + \alpha^k d^k) \leq f(x^k) - \gamma \left(\alpha^k \|d^k\|\right)^2,$$

which replaces the Armijo's condition

$$f(x^k + \alpha^k d^k) \leq f(x^k) + \gamma \alpha^k \nabla f(x^k)^\top d^k$$

Before describing the line search procedure, we state a formal assumption on the sequence of search directions.

Let $d^k \in R^n$ be a feasible direction at $x^k \in X$, where $X \subseteq \mathbb{R}^n$ is the feasible set. Let β^k be the maximum feasible step length along d^k. Taking, for instance, $\hat{x}^k \in X$ and $d^k = \hat{x}^k - x^k \neq 0$, it follows $\beta^k \geq 1$.

Assumption A.1. *$\{d^k\}$ is a sequence of feasible search directions such that*

(a) *for all k we have $\|d^k\| \leq M$ and $\beta^k \leq \bar{\beta}$ for given numbers $M > 0$ and $\bar{\beta} > 0$;*

(b) *for all k we have $\nabla f(x^k)^\top d^k < 0$.*

The Quadratic Line Search algorithm can be described as follows.

The properties of Algorithm QLS are stated in the next proposition.

Algorithm 12: Quadratic Line Search QLS

Input: $x^k \in X$, $d^k \in \mathbb{R}^n$, α_0^k, β^k
Output: α^k
Data: $\delta \in (0,1)$, $\gamma > 0$

if $\nabla f(x^k)^\top d^k \geq 0$ **then**
  return $\alpha^k \leftarrow 0$
end
$\alpha \leftarrow \min\{\alpha_0^k, \beta^k\}$;
$j \leftarrow 0$;
while $f(x^k + \alpha d^k) > f(x^k) - \gamma\left(\alpha \|d^k\|^2\right)$ **do**
  $\alpha \leftarrow \delta\alpha$;
  $j \leftarrow j + 1$;
end
return $\alpha^k \leftarrow \alpha$

Proposition A.2. *Let $\{x^k\}$ be a sequence of points belonging to the feasible set X, and let $\{d^k\}$ be a sequence of search directions satisfying Assumption* A.1. *Then:*

(i) *Algorithm* QLS *determines, in a finite number of iterations, a scalar α^k such that*

$$f(x^k + \alpha^k d^k) \leq f(x^k) - \gamma(\alpha^k \|d^k\|)^2; \tag{A.1}$$

(ii) *if $\{x^k\}$ converges to $\bar{x}$ and*

$$\lim_{k\to\infty} \left(f(x^k) - f(x^k + \alpha^k d^k)\right) = 0, \tag{A.2}$$

then we have

$$\lim_{k\to\infty} \alpha^k \|d^k\| = 0, \tag{A.3}$$

and

$$\lim_{k\to\infty} \beta^k \nabla f(x^k)^\top d^k = 0. \tag{A.4}$$

Proof. In order to prove assertion (i), let us assume, by contradiction, that condition of the while cycle is violated for every $j \in \{0, 1, \ldots\}$. Then, we have for $j = 0, 1, \ldots$

$$f(x^k + \bar{\alpha}\delta^j d^k) > f(x^k) - \gamma\left(\bar{\alpha}\delta^j\right)^2 \|d^k\|^2,$$

where $\bar{\alpha} = \min\{\beta^k, \lambda\}$. Taking limits for $j \to \infty$ we obtain:

$$\nabla f(x^k)^\top d^k \geq 0,$$

which contradicts (b) of Assumption A.1.

From (A.1) and (A.2) it follows that (A.3) holds. In order to prove (A.4), assume by contradiction that there exists an infinite subset $K \subseteq \{0, 1, \ldots\}$ such that

$$\lim_{k \in K, k \to \infty} \beta^k \nabla f(x^k)^\top d^k = \beta^\star \nabla f(\bar{x})^\top \bar{d} < 0. \tag{A.5}$$

From (A.5) we get, for $k \in K$ and k sufficiently large,

$$\beta^k \geq \eta_1, \tag{A.6}$$

and

$$\|d^k\| \geq \eta_2, \tag{A.7}$$

for some numbers $\eta_1, \eta_2 > 0$.

Then, (A.3) and (A.7) imply that we can write

$$\lim_{k \in K, k \to \infty} \alpha^k = 0. \tag{A.8}$$

From (A.8) and (A.6) we obtain, for $k \in K$ and k sufficiently large,

$$\alpha^k < \bar{\alpha} = \min\{\beta^k, \lambda\},$$

which implies, recalling the instructions of the algorithm,

$$f\left(x^k + \frac{\alpha^k}{\delta} d^k\right) > f(x^k) - \gamma \left(\frac{\alpha^k}{\delta} \|d^k\|\right)^2. \tag{A.9}$$

From (A.9), using the Mean Value Theorem, it follows

$$\nabla f(\xi^k)^\top d^k > -\gamma \left(\frac{\alpha^k}{\delta}\right) \|d^k\|^2, \tag{A.10}$$

where

$$\xi^k = x^k + t^k \frac{\alpha^k}{\delta} d^k$$

and $t^k \in (0, 1)$. Taking the limits for $k \in K$ and $k \to \infty$, recalling (A.8), we obtain

$$\nabla f(\bar{x})^\top \bar{d} \geq 0,$$

which contradicts (A.5). □

Appendix B

Publications

This research activity has led to several publications in international journals and conferences. These are summarized below. [1]

International Journals

1. D. Di Lorenzo, A. Galligari, and M. Sciandrone, "A convergent and efficient decomposition method for the traffic assignment problem", *Computational Optimization and Applications*, vol. 60, no. 1, pp. 151-170, 2015. [Online]. Available: `http://dx.doi.org/10.1007/s10589-014-9668-6`.
2. A. Galligari and M. Sciandrone, "A convergent and fast path equilibration algorithm for the traffic assignment problem", *Optimization Methods and Software*, pp. 1-18, 2017. [Online]. Available: `http://dx.doi.org/10.1080/10556788.2017.1332621`
3. A. Galligari, M. Maischberger, and F. Schoen, "Local search heuristics for the zone planning problem", *Optimization Letters*, vol. 11, no. 1, pp. 195-207, Jan 2017. [Online]. Available: `https://doi.org/10.1007/s11590-016-1069-6`.

National Conferences

1. A. Galligari, N. Bulgarini, and M. Sciandrone, "Decomposition algorithms for traffic assignment problems", Associazione Italiana di Ricerca Operativa (AIRO), Pisa (Italy), 7-10 Settembre 2015.

[1] The author's bibliometric indices are the following: H-index = 1, total number of citations = 3 (source: Google Scholar on October 2017).

Bibliography

[1] M. Abramson, C. Audet, G. Couture, J. Dennis, Jr., S. Le Digabel, and C. Tribes, "The NOMAD project," Software available at https://www.gerad.ca/nomad/. [Online]. Available: https://www.gerad.ca/nomad/

[2] G. Abrate, M. Piacenza, and D. Vannoni, "Regional Science and Urban Economics The impact of Integrated Tariff Systems on public transport demand : Evidence from Italy," *Regional Science and Urban Economics*, vol. 39, pp. 120–127, 2009.

[3] C. Audet and J. Dennis, Jr., "Mesh adaptive direct search algorithms for constrained optimization," *SIAM Journal on Optimization*, vol. 17, no. 1, 2006. [Online]. Available: http://dx.doi.org/doi:10.1137/040603371

[4] W. M. A. T. Authority, "http://http://www.wmata.com/."

[5] L. Babel and H. Kellerer, "Design of Tariff Zones in Public Transportation Networks : Theoretical Results and Heuristics," *Mathematical Methods of Operations Research (ZOR)*, vol. 58, pp. 4–5, 2003.

[6] H. Bar-Gera, "Origin-Based Algorithm for the Traffic Assignment Problem," *Transportation Science*, vol. 36, pp. 398–417, 2002.

[7] ——, "Traffic assignment by paired alternative segments," *Transportation Research Part B: Methodological*, vol. 44, no. 8-9, pp. 1022–1046, sep 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0191261509001350

[8] W. M. A. T. A. Blog, "http://planitmetro.com/2012/10/31/data-download-metrorail-ridership-by-origin-and-destination/."

[9] M. Brenninger-GÃ¶the, K. O. JÃ¶rnsten, and J. T. Lundgren, "Estimation of origin-destination matrices from traffic counts using multiobjective programming formulations," *Transportation Research Part B: Methodological*, vol. 23, no. 4, pp. 257 – 269, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0191261589900283

[10] A. Chen and R. Jayakrishnan, *A path-based gradient projection algorithm: effects of equilibration with a restricted path set under two flow update policies*, ser. Working paper. Institute of Transportation Studies, University of California, 1998. [Online]. Available: https://books.google.it/books?id=0m0nAQAAMAAJ

[11] S. C. Dafermos and F. T. Sparrow, "The traffic assignment problem for a general network," *Journal of Research of the National Bureau of Standards, Series B*, vol. 73, no. 2, pp. 91–118, 1969.

[12] D. Di Lorenzo, A. Galligari, and M. Sciandrone, "A convergent and efficient decomposition method for the traffic assignment problem," *Computational Optimization and Applications*, vol. 60, no. 1, pp. 151–170, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10589-014-9668-6

[13] R. B. Dial, "A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration," *Transportation Research Part B: Methodological*, vol. 40, no. 10, pp. 917–936, dec 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0191261506000269

[14] G. Dos Reis and B. Stroustrup, "Specifying C++ concepts," *SIGPLAN Not.*, vol. 41, pp. 295–308, January 2006.

[15] C. S. Fisk, "Trip matrix estimation from link traffic counts: The congested network case," *Transportation Research Part B*, vol. 23, no. 5, pp. 331–336, oct 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/019126158990009X

[16] M. Florian, I. Constantin, and D. Florian, "A New Look at Projected Gradient Method for Equilibrium Assignment," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2090, no. March 2016, pp. 10–16, 2009. [Online]. Available: http://trrjournalonline.trb.org/doi/10.3141/2090-02

[17] M. Florian, M. Gaudry, and C. Lardinois, "A two-dimensional framework for the understanding of transportation planning models," *Transportation Research Part B: Methodological*, vol. 22, no. 6, pp. 411–419, 1988. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0191261588900227

[18] M. Florian and D. Hearn, "Network Equilibrium Models and Algorithms," *Handbooks in Operations Research and Management Science*, vol. 8, pp. 485–550, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0927050705801100

[19] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956. [Online]. Available: http://dx.doi.org/10.1002/nav.3800030109

[20] M. Fukushima, "A modified frank-wolfe algorithm for solving the traffic assignment problem," *Transportation Research Part B: Methodological*, vol. 18, no. 2, pp. 169 – 177, 1984. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0191261584900298

[21] A. Galligari, M. Maischberger, and F. Schoen, "Local search heuristics for the zone planning problem," *Optimization Letters*, vol. 11, no. 1, pp. 195–207, Jan 2017. [Online]. Available: https://doi.org/10.1007/s11590-016-1069-6

[22] A. Galligari and M. Sciandrone, "A convergent and fast path equilibration algorithm for the traffic assignment problem," *Optimization Methods and Software*, vol. 0, no. 0, pp. 1–18, 2017. [Online]. Available: http://dx.doi.org/10.1080/10556788.2017.1332621

[23] U. M. García-Palomares, I. J. García-Urrea, and P. S. Rodríguez-Hernández, "On sequential and parallel non-monotone derivative-free algorithms for box constrained optimization," *Optimization Methods and Software*, vol. 28, no. 6, pp. 1233–1261, 2013.

[24] M. Gendreau and J.-Y. Potvin, Eds., *Handbook of Metaheuristics*, 2nd ed., ser. International Series in Operations Research and Management Science. Springer, 2010, vol. 146.

[25] G. Gentile, "Local user cost equilibrium: a bush-based algorithm for traffic assignment," *Transportmetrica A: Transport Science*, vol. 10, no. 1, pp. 15–54, 2014.

[26] F. W. Glover and M. Laguna, *Tabu Search*. Springer, June 1998.

[27] H. W. Hamacher and A. SchÃ¶bel, "Design of zone tariff systems in public transportation," *Operations Research*, vol. 52, no. 6, pp. 897–908, 2004.

[28] R. Jayakrishnan, W. K. Tsai, J. Prashker, and S. Rajadhyaksha, "A Faster Path-based Algorithm for Traffic Assignment," *Transportation Research Record*, vol. 1443, pp. 75–83, 1994.

[29] M. Josefsson and M. Patriksson, "Sensitivity analysis of separable traffic equilibrium equilibria with application to bilevel optimization in network design," *Transportation Research Part B: Methodological*, vol. 41, no. 1, pp. 4–31, 2007.

[30] B. Kostic and G. Gentile, "Using traffic data of various types in the estimation of dynamic O-D matrices," *2015 International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2015*, no. June, pp. 66–73, 2015.

[31] P. Kowalik, "An improvement of a "price–oriented" public transport tariff system," Wydawnictwo Akademii Techniczno-Humanistycznej, Bielsko-Biala, Tech. Rep., 2010.

[32] T. Larsson and M. Patriksson, "Simplicial Decomposition with Disaggregated Representation for the Traffic Assignment Problem," *Transportation Science*, vol. 26, no. 1, pp. 4–17, 1992.

[33] S. Le Digabel, "Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm," *ACM Transactions on Mathematical Software*, vol. 37, no. 4, pp. 1–15, 2011.

[34] L. J. LeBlanc, R. V. Helgason, and D. E. Boyce, "Improved efficiency of the frank-wolfe algorithm for convex network programs," *Transportation Science*, vol. 19, no. 4, pp. 445–462, 1985. [Online]. Available: https://doi.org/10.1287/trsc.19.4.445

[35] T. Leventhal, Nemhauser, "A column generation algorithm for optimal traffic assignment," *Transportation Science*, vol. 7, no. 2, pp. 168–176, 1973.

[36] C.-J. Lin, S. Lucidi, L. Palagi, A. Risi, and M. Sciandrone, "Decomposition algorithm model for singly linearly-constrained problems subject to lower and upper bounds," *Journal of Optimization Theory and Applications*, vol. 141, no. 1, pp. 107–126, 2009.

[37] G. Liuzzi and A. Risi, "A decomposition algorithm for unconstrained optimization problems with partial derivative information," *Optimization Letters*, vol. 6, no. 3, pp. 437–450, 2012.

[38] H. Lourenço, O. Martin, and T. Stülze, "Iterated Local Search," in *Handbook of Metaheuristics*, ser. International Series in Operations Research & Management Science. Kluwer Academic Publisher, 2003, vol. 57, ch. 11, pp. 321–353.

[39] M. Maischberger, "COIN-OR METSlib: a metaheuristics framework in modern C++," Optimization Online, Tech. Rep., 2011.

[40] M. Manheim, *Fundamentals of transportation systems analysis*, ser. Fundamentals of Transportation Systems Analysis, 1976, no. v. 1. [Online]. Available: https://books.google.it/books?id=WGwgAQAAMAAJ

[41] M. G. Mcnally, "The Four Step Model," *Transportation*, no. UCI-ITS-AS-WP-00-51, pp. 35–52, 2007. [Online]. Available: http://www.escholarship.org/uc/item/0r75311t

[42] Q. Meng, H. Yang, and M. G. Bell, "An equivalent continuously differentiable model and a locally convergent algorithm for the continuous network design problem," *Transportation Research Part B: Methodological*, vol. 35, no. 1, pp. 83–105, jan 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0191261500000163

[43] M. Mitradjieva and P. O. Lindberg, "The Stiff Is Moving - Conjugate Direction Frank-Wolfe Methods with Applications to Traffic

Assignment *," *Transportation Science*, vol. 47, no. 2, pp. 280–293, 2013. [Online]. Available: http://pubsonline.informs.org/doi/abs/10.1287/trsc.1090.0306$\delimiter"026E30F$nhttp://pubsonline.informs.org/doi/abs/10.1287/trsc.1120.0409

[44] Y. Nie, "A Note on Bar-Gera's Algorithm for the Origin-Based Traffic Assignment Problem," *Transportation Science*, vol. 46, no. 1, pp. 27–38, 2012.

[45] Y. M. Nie, "A class of bush-based algorithms for the traffic assignment problem," *Transportation Research Part B: Methodological*, vol. 44, no. 1, pp. 73–89, jan 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0191261509000769

[46] Y. M. Nie and H. M. Zhang, "A relaxation approach for estimating origin–destination trip tables," *Networks and Spatial Economics*, vol. 10, no. 1, pp. 147–172, Mar 2010. [Online]. Available: https://doi.org/10.1007/s11067-007-9059-y

[47] OpenMP Architecture Review Board, "OpenMP C and C++ application program interface," march 2002.

[48] OpenStreetMap, "http://www.openstreetmap.org."

[49] J. d. D. Ortuzar and L. G. Willumsen, *Modelling Transport*, 4th ed. Wiley, 2011.

[50] M. Patriksson, *The traffic assignment problem: models and methods*, ser. Topics in transportation. VSP, 1994. [Online]. Available: https://books.google.it/books?id=eoQnAQAAMAAJ

[51] ——, "Algorithms for computing traffic equilibria," *Networks and Spatial Economics*, vol. 4, no. 1, pp. 23–38, Mar 2004. [Online]. Available: https://doi.org/10.1023/B:NETS.0000015654.56554.31

[52] O. Perederieieva, M. Ehrgott, A. Raith, and J. Y. Wang, "A framework for and empirical study of algorithms for traffic assignment," *Computers & Operations Research*, vol. 54, pp. 90–107, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054814002354

[53] S. Powell, "The convergence of equilibrium algorithms with predetermined step sizes," *Transportation Science*, vol. 16, no. 1, pp. 45–55, 1982.

[54] J. B. Rosen, "The gradient projection method for nonlinear programming. part i. linear constraints," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 1, pp. 181–217, 1960. [Online]. Available: http://www.jstor.org/stable/2098960

[55] S. Ryu, A. Chen, and K. Choi, "A modified gradient projection algorithm for solving the elastic demand traffic assignment problem," *Computers*

& Operations Research, vol. 47, pp. 61–71, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305054814000252

[56] N. Sang, *Estimating an OD matrix from network data : a network equilibrium approach.* Montreal : Universite de Montreal, Centre de Recherche sur les Transports, 1977, includes bibliographical references.

[57] A. Schöbel, *Optimization in public transportation.* Springer, 2006, ch. Tariff planning, pp. 207–236.

[58] Y. Sheffi, "Urban transportation networks," 1985.

[59] W. Shen and L. Wynter, "A new one-level convex optimization approach for estimating origin-destination demand," *Transportation Research Part B: Methodological*, vol. 46, no. 10, pp. 1535–1555, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.trb.2012.07.005

[60] R. Stallman, *Using GCC: the GNU compiler collection reference manual.* Boston, MA, USA: GNU Press, 2003.

[61] H. J. Van Zuylen and L. G. Willumsen, "The most likely trip matrix estimated from traffic counts," *Transportation Research Part B: Methodological*, vol. 14, no. 3, pp. 281–293, sep 1980. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0191261580900089

[62] J. Wardrop, "Some theoretical aspects of road traffic research communication networks," *Proceedings - Institution of Civil Engineers 2*, vol. 2, pp. 325–378, 1952.

[63] H. Yang, "Heuristic algorithms for the bilevel origin-destination matrix estimation problem," *Transportation Research Part B: Methodological*, vol. 29, no. 4, pp. 231–242, 1995.

[64] H. Zheng, "Adaptation of Network Simplex for the Traffic Assignment Problem," *Transportation Science*, vol. 49, no. 3, pp. 543–558, 2015.