Decomposition methods based on projected gradient for network equilibrium problems

A. Cassioli [*], D. Di Lorenzo[*], M. Sciandrone[*]

Abstract

In this work we consider the symmetric network equilibrium problem formulated as convex minimization problem whose variables are the path flows. In order to take into account the difficulties related to the large dimension of real network problems we adopt a column generation strategy and we employ a gradient projection method within an inexact decomposition framework. We present a general decomposition algorithm model and we derive several specific algorithms for network equilibrium problems. Global convergence results are established. Computational experiments performed on medium-large dimension problems show the validity and the effectiveness of the proposed approach.

Keywords: decomposition method, network equilibrium, projected gradient, column generation, Gauss-Seidel method.

1 Introduction

Network assignment problems are a widely studied subject in many research fields, as for instance transportation and data transmission. The aim of a network equilibrium model is to predict the link flows of a network which depend

[*]Università degli Studi di Firenze, Via di S.Marta 3, 50139 - Firenze (IT)

on the routes origin/destination chosen by the users (travellers or data package) of the network.

The network is model by a direct graph, whose nodes represent origins, destinations, and intersections, and arcs represent the transportation links. There is a set of node pairs, called Origin/Destination (OD), and for each OD pair there is a known demand. For each link there is a user cost function depending, in general, on the link flow of the whole graph. Every infinitesimal unit of flow travels from its origin to its destination along a path that minimizes its own travel cost, so the network will eventually reach stability in a Nash equilibrium point. The Wardrop's user-optimal principle states that, if the network is in equilibrium, then all the routes used by the users have a cost less or equal to that of any unused route. The Wardrop equilibrium conditions lead to solve a variational inequality which, under suitable assumptions on the cost functions, is equivalent to a convex optimization problem. For instance, the equivalence between the variational inequality and the convex optimization problem holds whenever the cost f_a of any given link a is non-decreasing and depends only on the flow y_a through the given link. We address the reader to [6] for the technical details and the assumptions which lead to the convex optimization problems object of this work (see below).

Let P be the number of OD pairs, n the number of paths between all the origins and all the destinations, and n_p the number of paths between the origin and destination of the p-th OD pair, so that we have $n = n_1 + n_2 + \ldots + n_P$. We denote by $x \in \mathbb{R}^n$ the vector of path flows.

We will denote by bracketed subscripts the subvectors, i.e. $x_{(h)}, h \in \{1, \ldots, P\}$ will denote the variables of x whose indexes are in h. To ease the notation, no brackets are used if h is a singleton.

We partition the vector of variables x as follows

$$x = \left(x_{(1)}, \ldots, x_{(p)}, \ldots, x_{(P)}\right)^T,$$

where $x_{(p)} \in \mathbb{R}^{n_p}$, $p \in \{1, \ldots, P\}$, and we introduce two convex minimization problems as equivalent formulations of symmetric network equilibrium problems. The two formulations lead to algorithms operating in the space of arc flow and in the space of path flow, respectively. We remark that real network equilibrium problems are very large scale problems, and this is the main issue to be considered in the design of optimization algorithms.

Let m be the number of arcs of the graph. We denote with $y \in \mathbb{R}^m$ the arc flow vector. Arc and path flows are related by $y = Hx$, where $H \in \mathbb{R}^{m \times n}$ is the arc-path incidence matrix whose generic element h_{ij} is equal to 1 if arc i belongs to path j and is equal to 0 otherwise. The optimization problem with arc variables takes the structure

$$\begin{aligned} \min_{x,y} \quad & F(y) \\ & y = Hx \\ & e^T x_{(p)} = d_p \quad \forall p \in \{1, \ldots, P\} \\ & x_{(p)} \geq 0 \qquad \forall p \in \{1, \ldots, P\} \end{aligned} \tag{1}$$

where $F : R^m \to R$ is separable, i.e., $F(y) = \sum_{i=1}^m F_i(y_i)$, and each $F_i : \mathbb{R}^+ \to \mathbb{R}^+$, is a convex continuously differentiable function; d_p is the demand of the p-th OD pair.

Traditional approaches for symmetric network equilibrium problems are arc-based algorithms using formulation (1). The most used arc-based algorithm is

the Frank-Wolfe algorithm, since it requires to store only arc flows (so it is suitable for solving real problems) and is very simple to implement; indeed, it is sufficient to compute the shortest paths for all the OD pairs to determine, at any iteration, the search direction. Recently, arc flow-based algorithms have been proposed in [4, 13]. Despite the astonishing performance both in terms of memory usage and execution time, arc flow-based algorithms have as drawback the lack of information about the actual paths followed by the network users. Indeed, knowing how exactly the flow demand distributes along the network is often of great use, and this is one of the reasons motivating the design of path-flow algorithms for the solution of the path-based optimization problem described below and object of the present work.

By simple substitution in (1), we can consider the following equivalent path-based optimization problem

$$\begin{aligned} &\min_x f(x) = F(Hx) \\ &x \in \mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2 \times \ldots \times \mathcal{F}_P \end{aligned} \tag{2}$$

where we denote by $\mathcal{F}_p$ the set

$$\mathcal{F}_p = \left\{ x \in \mathbb{R}^{n_p} : e^T x_{(p)} = d_p, \quad x_{(p)} \geq 0 \right\} \tag{3}$$

The convex problem (2) has a very simple structure, as its feasible set $\mathcal{F}$ is the Cartesian product of simplices. However, problem (2) can be considered a "virtual" formulation: indeed, in any real application it is not reasonable to completely enumerate, a priori, all the paths, since this would be too expensive.

In order to take into account the difficulties related to the large dimension of problem (2) we adopt a standard column generation strategy (by iteratively adding only the variables, i.e., the paths, of the model needed to reach optimal-

ity) and we employ a gradient projection method within an inexact decomposition framework. Similar approaches have been proposed in [15] for solving the asymmetric traffic equilibrium problem formulated as a variational inequality, and in [5, 9] for solving the symmetric traffic equilibrium problem formulated as (2). In particular, in [5] an adaptation of the Rosen's projected gradient algorithm is used within a theoretically exact decomposition Gauss-Seidel scheme.

The paper is organized as follows: in Section 2, with reference to a general problem defined on the Cartesian product of convex sets, we present an inexact decomposition algorithm using restricted feasible sets (belonging to lower dimensional spaces to possibly tackle large dimensional problems), and gradient projection iterations. Under suitable assumptions on the restricted feasible sets, we prove the global convergence of the presented algorithm. The specific case of traffic equilibrium problem is the topic of Section 3, in which different decomposition schemes are derived from the general framework previously defined. Computational results are presented in Section 4. Conclusions and future directions are summarized in Section 5. In Appendix A we recall known properties of Armijo line search, while in Appendix B we prove some results used in our convergence analysis.

2 Inexact decomposition algorithm using restricted feasible sets

Let us consider the problem

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ &x \in \mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2 \times \ldots \times \mathcal{F}_L \end{aligned} \tag{4}$$

where $f : R^n \to R$ is a convex continuously differentiable function, $\mathcal{F}_h \subseteq \mathbb{R}^{n_h}$, $h = 1, \ldots, L$, are compact convex sets, and $n_1 + \ldots + n_h + \ldots + n_L = n$. Taking into account the structure of the feasible set $\mathcal{F}$, we partition the vector of variables x as follows

$$x = \left(x_{(1)}, \ldots, x_{(h)}, \ldots, x_{(L)}\right)^T,$$

where $x_{(h)} \in \mathbb{R}^{n_h}$, $h = 1, \ldots, L$.

Given a point $y \in \mathcal{F}$, for $h \in \{1, \ldots, L\}$, we denote with by $\mathcal{F}_h(y)$ a closed convex set depending on the point y and such that $\mathcal{F}_h(y) \subseteq F_h$. Formally, we state the following assumption on the sets $F_h(\cdot)$ for $h \in \{1, \ldots, L\}$.

Assumption 1.

(i) *The set* $\cup_{x \in \mathcal{F}} \{\mathcal{F}_h(x)\}$ *has finite cardinality.*

(ii) *Let* $K \subseteq \{0, 1, \ldots, \}$ *be an infinite subsequence such that* $x^k \in \mathcal{F}$ *for all* $k \in K$*, and assume that* $x^k \to \bar{x}$ *for* $k \in K$ *and* $k \to \infty$*. If for* $k \in K$ *and* k *sufficiently large we have that*

$$\bar{x}_{(h)} \in \arg \min_{x_{(h)} \in \mathcal{F}_h(x^k)} f(\bar{x}_{(1)}, \ldots x_{(h)}, \ldots, \bar{x}_{(L)}),$$

then it holds

$$\bar{x}_{(h)} \in \arg \min_{x_{(h)} \in \mathcal{F}_h} f(\bar{x}_{(1)}, \ldots x_{(h)}, \ldots, \bar{x}_{(L)}).$$

We observe that (i) of Assumption 1 is a technical condition needed to manage projection operations for infinite subsequences. Condition (ii) requires that the restricted sets $\mathcal{F}_h(x^k)$ capture, in the limit, the geometry of the set $\mathcal{F}_h$ in terms of optimality conditions. Just to have an idea how to satisfy condition (ii), we can consider the case (central in the present work) of $\mathcal{F}_h$ defined as a

simplex, that is

$$\mathcal{F}_h = \left\{ x \in \mathbb{R}^{n_h} : e^T x_{(h)} = d_h, \quad x_{(h)} \geq 0 \right\}.$$

Given a feasible point x^k, the restricted set $\mathcal{F}_h(x^k)$ can be defined by considering as variables only the components $x^k_{(h),i} > 0$ and that component $x^k_{(h),i^\star} = 0$ that mostly violates the optimality conditions, that is,

$$\left\{ \frac{\partial f(x^k)}{x_{(h),i^\star}} \right\} \leq \left\{ \frac{\partial f(x^k)}{x_{(h),j}} \right\} \qquad j = 1, \ldots, n_h.$$

We will show later (see Appendix B) that the above definition of the restricted set $\mathcal{F}_h(x^k)$ allows us to satisfy Assumption 1.

Now we describe the inexact decomposition algorithm based on the gradient projection and on the well-known Armijo-type line search (whose details are reported in Appendix A), together with a theoretical result employed in our convergence analysis.

In order to ensure the global convergence of Algorithm IDA we need to introduce the following assumption, which requires that each index $l \in \{1, \ldots, L\}$ (corresponding to the block component $x_{(l)}$) is periodically considered at Step 1 within a prefixed maximum number of iterations.

Assumption 2. *There exists an integer $M > 0$ such that, for all $k \geq 0$ and for all $l \in \{1, \ldots, L\}$, we can find an index $l(k)$, with $0 \leq l(k) \leq M$, such that at Step 1 we have $h^{k+l(k)} = l$.*

Note that setting, for instance, $h^k = (k \mod L) + 1$, Algorithm IDA reduces to an inexact Gauss-Seidel algorithm (on restricted feasible component subsets) where a single iteration of the projection gradient method is performed for every block-component. The literature on the convergence of exact decomposition algorithms is wide (see, eg., [2], [7], [10], [11]). A recent study on inexact Gauss-

Algorithm 1: Inexact Decomposition Algorithm (IDA)

Input: $x^0 \in \mathcal{F}$

1 $k \leftarrow 0$;

2 choose $h^k \in \{1, \dots, L\}$;

3 define $\mathcal{F}_{h^k}(x^k)$;

4 set

$$d^k_{(i)} = \begin{cases} 0_{(i)} & \text{if } i \neq h^k \\ \hat{x}^k_{(i)} - x^k_{(i)} & \text{if } i = h^k \end{cases}$$

where $\hat{x}^k_{(h^k)} = P_{\mathcal{F}_{h^k}(x^k)}[x^k_{(h)} - \nabla_{(h)} f(x^k)]$;

5 set

$$x^{k+1} = x^k + \alpha^k d^k,$$

where α^k is computed by means of the Armijo line search;

6 $k \leftarrow k + 1$;

7 go to 2;

Seidel algorithms based on gradient projection mappings has been performed in [3]. We remark that the convergence analysis of Algorithm IDA can not be derived from results stated in preceding works, since Algorithm IDA involves restricted feasible subsets $\mathcal{F}_h(x^k)$ instead of the prefixed subsets $\mathcal{F}_h$.

We are ready to state the following convergence result.

Proposition 1. *Let $\{x^k\}$ be the sequence generated by the Algorithm* IDA. *Suppose that for all k the sets $\mathcal{F}_{h^k}(x^k)$ defined at Step 1 satisfy Assumption* 1, *and that the sequence $\{h^k\}$ is such that Assumption* 2 *holds. Then $\{x^k\}$ admits limit points and each limit point is a solution of problem* (2).

Proof. The points of the sequence $\{x^k\}$ belong to the feasible compact set, so $\{x^k\}$ admits limit points.

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

$$\lim_{k \in K, k \to \infty} x^k = x^\star. \tag{5}$$

The instructions of the algorithm imply

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

so that, as f is bounded below, we can write

$$\lim_{k\to\infty} f(x^{k+1}) - f(x^k) = 0. \tag{6}$$

From the properties of the projection mapping we get

$$\nabla f(x^k)^T d^k = \nabla_{(h^k)} f(x^k)^T d^k_{(h^k)} \leq -\|\hat{x}^k_{(h^k)} - x^k_{(h^k)}\|^2 = -\|\hat{x}^{k+1} - x^k\|^2, \tag{7}$$

being

$$\hat{x}^k_{(h^k)} = P_{\mathcal{F}_{h^k}(x^k)}[x^k_{(h^k)} - \nabla_{(h^k)} f(x^k)]. \tag{8}$$

Furthermore, for all $k \in K$ we have

$$f(x^{k+1}) \leq f(x^k) + \gamma \alpha^k \nabla f(x^k)^T d^k,$$

where α^k is determined by means of the Armijo line search along the search direction d^k. Note that, due to the convexity of $\mathcal{F}_{h^k}(x^k)$, the maximum feasible step length β^k along d^k is greater than or equal to 1. Furthermore, as the closed convex set $\mathcal{F}_{h^k}(x^k)$ belongs, by assumption, to the compact set $\mathcal{F}$, we have that the search direction d^k is bounded.

Using (6) and assertion (ii) of Proposition 8 we obtain

$$\lim_{k\to\infty} \nabla f(x^k)^T d^k = 0. \tag{9}$$

From (9) and (7) it follows

$$\lim_{k\to\infty} \|x^{k+1} - x^k\| = \lim_{k\to\infty} \|\hat{x}^k_{(h^k)} - x^k_{(h^k)}\| = \lim_{k\to\infty} \|d^k\| = 0. \tag{10}$$

From (i) of Assumption 1 it follows that, for every $j \in \{1, \ldots, L\}$, we can find an infinite subset $K_1 \subseteq K$ and an $\mathcal{F}_j^\star$ such that

$$\mathcal{F}_j(x^k) = \mathcal{F}_j^\star \qquad \forall k \in K_1.$$

Recalling Assumption 2 we have that $h^{k+j(k)} = j$, with $0 \leq j(k) \leq M$, and hence, using (10) we can write

$$\lim_{k\in K_1, k\to\infty} x^{k+j(k)} = x^\star. \tag{11}$$

From (8), (10) and (11), recalling the continuity of the projection mapping, we obtain

$$x^\star_{(j)} = P_{\mathcal{F}_j^\star}[x^\star_{(j)} - \nabla_{(j)} f(x^\star)], \tag{12}$$

which implies that

$$x^\star_{(j)} \in \arg\min_{x_{(j)}\in\mathcal{F}_j^\star} f(x^\star_{(1)}, \ldots, x_{(j)}, \ldots, x^\star_{(L)}). \tag{13}$$

Taking into account (13) and recalling (ii) of Assumption 1 we have

$$x^\star_{(j)} \in \arg\min_{x_{(j)}\in\mathcal{F}_j} f(x^\star_{(1)}, \ldots, x_{(j)}, \ldots, x^\star_{(L)})$$

This equation holds for every $j \in \{1, \ldots, L\}$, and hence the proposition is proved. □

3 Decomposition algorithms for NEP

In this section we present two decomposition schemes for the special case of the network equilibrium problem defined by (2) and (3), as well as the case with no decomposition at all. We recall that

- given $x \in \mathcal{F}$, $x_{(h),i}$ is the flow of the i-th path between the h-th OD pair;
- $f : R^n \to R$ is a convex continuously differentiable function;
- the partial derivative

$$\frac{\partial f(x)}{\partial x_{(h),i}}$$

  is the cost of the i-th path.

We exploit the special structure of the feasible set to derive two algorithms from the general framework depicted in Algorithm 1. To this aim we need to specify:

(a) the rule for constructing the restricted feasible set $\mathcal{F}_h(x)$;

(b) the rule for selecting the block variables $x_{(h^k)}$ to be updated at each iteration k.

3.1 OD-pairs based decomposition algorithm

Problem (2) has the same form of problem (4), being $L = P$ the number of convex compact subsets $\mathcal{F}_h$ whose Cartesian product defines the feasible set $\mathcal{F}$. Formulation (2) naturally leads to an *OD-pair based decomposition* scheme.

Concerning point (a), since each subproblem has a huge number of variables (i.e., there exists a huge number of paths between each OD pair), the aim is to avoid of enumerating them a priori. Thus, the basic idea is to consider, for each

subproblem, only the variables whose current value is strictly positive (corresponding to paths carrying non-zero flows), and the variable that corresponds to the cheaper path to the destination. In this way we consider restricted feasible sets belonging to lower dimensional subspaces.

Formally, given $\bar{x} \in \mathcal{F}$, we set

$$I_h^+(\bar{x}) = \{i \in \{1, \ldots, n_h\} : \ \bar{x}_{(h),i} > 0\}$$

and

$$\pi_h(\bar{x}) \in \arg \min_{j=1,\ldots,n_h} \left\{ \frac{\partial f(\bar{x})}{x_{(h),j}} \right\}.$$

Then we define the index set identifying the variables to be updated

$$I_h(\bar{x}) = I_h^+(\bar{x}) \cup \pi_h(\bar{x}).$$

For each $h \in \{1, \ldots, P\}$ we introduce the restricted feasible set $\mathcal{F}_h(\bar{x})$ defined as follows

$$\mathcal{F}_h(\bar{x}) = \{x_{(h)} \in \mathbb{R}^{n_h} : x_{(h)} \in \mathcal{F}_h, \ x_{(h),i} = 0 \ \forall i \notin I_h(\bar{x})\}. \tag{14}$$

Assumption 1 holds for the restricted feasible set defined by (14), as stated in the next proposition whose proof is reported in Appendix B.

Proposition 2. *For any point $x \in \mathcal{F}$ and for $h = 1, \ldots, P$ let $\mathcal{F}_h(x)$ be the restricted feasible set defined as in* (14)*. Then Assumption* 1 *holds.*

Note that the definition of the set $\mathcal{F}_h(\bar{x})$ requires to determine the index $\pi_h(\bar{x})$, and this can be done by computing the shortest path between the considered origin and destination.

As regards point (b), we keep h_k constant for a fixed number of iterations $n_{iter} \geq 1$, that is, each block of variables $x_{(h)}$ is sequentially selected and n_{iter}

iterations of the gradient projection method are performed for its updating.

The formal description of the algorithm, which is a specific realization of Algorithm IDA, named IDA-OD, is reported in Algorithm 2.

Algorithm 2: Inexact Decomposition Algorithm for Origin-Destination pairs (IDA-OD)

Input: $x^0 \in \mathcal{F}$, $n_{iter} \geq 1$.

1 $k \leftarrow 0$, $l \leftarrow 1$, $count \leftarrow 0$;

2 $h^k \leftarrow l$;

3 define $\mathcal{F}_{h^k}(x^k)$ as in (14) replacing $\bar{x}$ with x^k;

4 set

$$d^k_{(i)} = \begin{cases} 0_{(i)} & \text{if } i \neq h^k \\ \hat{x}^k_{(i)} - x^k_{(i)} & \text{if } i = h^k \end{cases}$$

where $\hat{x}^k_{(h^k)} = P_{\mathcal{F}_{h^k}(x^k)}[x^k_{(h)} - \nabla_{(h)} f(x^k)]$;

5 set

$$x^{k+1} = x^k + \alpha^k d^k,$$

where the stepsize α^k is computed by means of the Armijo line search;

6 **if** $count < n_{iter}$ **then**

7 $\quad count \leftarrow count + 1$

8 **else**

9 $\quad count \leftarrow 0, l \leftarrow (l \mod P) + 1$

10 **end**

11 $k \leftarrow k + 1$;

12 go to 2;

The global convergence of the algorithm follows from Proposition 1 and is stated in the following proposition.

Proposition 3. *Let* $\{x^k\}$ *be the sequence generated by the Algorithm* IDA-OD. *Then* $\{x^k\}$ *admits limit points and each limit point is a solution of problem* (2).

Proof. The rule for defining the restricted feasible set $\mathcal{F}_{h^k}(x^k)$ at Step 1 is such that Assumption 1 holds (see Proposition 2).

The sequence $\{h^k\}$ is generated in such a way that Assumption 2 is satisfied with $M = L \cdot n_{iter}$.

Then the thesis follows from Proposition 1.

□

3.2 Origin based decomposition algorithm

The network equilibrium problem formulated in (2) can be rewritten in an equivalent form in order to define another decomposition scheme.

More specifically, denoting by O the set of the origins, with $|O| = R$, and denoting with $D(h)$, for each $h \in O$, the set of destinations associated to the origin h, we can equivalently write the problem as follows

$$\begin{aligned} &\min_x f(x) \\ &x \in \mathcal{F} = \Omega_1 \times \Omega_2 \times \ldots \Omega_R \end{aligned} \tag{15}$$

where

$$\Omega_h = \prod_{l \in D(h)} \mathcal{F}_l.$$

Formulation (15) induces an *Origin-based decomposition* scheme.

Again, concerning point (a), following the strategy previously described for the OD-pair based decomposition algorithm, given $\bar{x} \in \mathcal{F}$, for each $h \in O$ we introduce the restricted feasible set $\Omega_h(\bar{x})$ defined as follows

$$\Omega_h(\bar{x}) = \prod_{l \in D(h)} \mathcal{F}_l(\bar{x}), \tag{16}$$

where the restricted feasible subset $\mathcal{F}_l(\bar{x})$ is defined in (14). We can prove that the restricted feasible set so defined is such that Assumption 1 holds as stated in the next proposition whose proof is reported in Appendix B.

Proposition 4. *For any point $x \in \mathcal{F}$ and for $h = 1, \ldots, R$ let $\Omega_h(x)$ be the*

restricted feasible set defined as in (16). *Then Assumption* 1 *holds.*

Note that the definition of the set $\Omega_h(\bar{x})$ requires to determine the indexes $\pi_l(\bar{x})$, for $l \in D(h)$ and this can be done by computing the shortest paths tree between the considered origin and all its destinations.

The rule for selecting the block variables $x_{(h^k)}$ to be updated at each iteration k is the same adopted for Algorithm IDA-OD.

We do not yield the formal description the algorithm, called IDA-O, since it can be immediately derived from the previously described Algorithm IDA-OD by replacing the restricted feasible set $\mathcal{F}_{h^k}(x^k)$ with $\Omega_{h^k}(x^k)$.

The global convergence of the algorithm follows from Proposition 1 and is stated in the following proposition.

Proposition 5. *Let* $\{x^k\}$ *be the sequence generated by the Algorithm* IDA-O. *Then* $\{x^k\}$ *admits limit points and each limit point is a solution of problem* (15).

Proof. The rule for defining the restricted feasible set $\Omega_{h^k}(x^k)$ at Step 1 is such that Assumption 1 holds (see Proposition 4).

The sequence $\{h^k\}$ is generated in such a way that Assumption 2 is satisfied with $M = R * n_{iter}$.

Then the thesis follows from Proposition 1.

□

3.3 Column-generation based Projected Gradient

The network equilibrium problem formulated in (2) can be in also tackled without using a decomposition scheme at all. Indeed, the projected gradient framework based on the column-generation approach directly applies when no decomposition is used, i.e. a special case in which we select all OD pairs at each iteration.

For what concern point (a), following the strategy previously described for the OD-pair based decomposition algorithm, given $\bar{x} \in \mathcal{F}$, the restricted feasible set $\Gamma(\bar{x})$ is defined as follows

$$\Gamma(\bar{x}) = \prod_{p \in P} \mathcal{F}_p(\bar{x}), \tag{17}$$

where the restricted feasible subset $\mathcal{F}_p(\bar{x})$ is defined in (14). Again, we can prove that $\Gamma(\bar{x})$ is such that Assumption 1 holds as stated in the next proposition whose proof is reported in Appendix B.

Proposition 6. *For any point $x \in \mathcal{F}$ let $\Gamma(x)$ be the restricted feasible set defined as in* (17)*. Then Assumption* 1 *holds.*

Note that the definition of the set $\Gamma(\bar{x})$ requires to determine the indexes $\pi_p(\bar{x})$, for $p \in P$ and this can be done by computing the shortest paths tree between all origins to all their destinations.

The resulting algorithm, denoted as PG, can be obtained from Algorithm IDA-OD by replacing the restricted feasible set $\mathcal{F}_{h^k}(x^k)$ with $\Gamma(x^k)$.

The global convergence of the algorithm follows from Proposition 1 and is stated in the following proposition.

Proposition 7. *Let $\{x^k\}$ be the sequence generated by the Algorithm* PG*. Then $\{x^k\}$ admits limit points and each limit point is a solution of problem* (15)*.*

Proof. The rule for defining the restricted feasible set $\Gamma(x^k)$ at Step 1 is such that Assumption 1 holds (see Proposition 4). Assumption 2 trivially holds, since we select all OD pairs at each iteration.

Then the thesis follows from Proposition 1. □

4 Numerical experiments

In this section we show the results of the computational experiments performed by the proposed inexact decomposition algorithms based on a column generation strategy. The aims of the experimentation were mainly the following:

(a) to verify the applicability of the column generation-based strategy (characterizing the new methods) for solving real large dimensional problems;

(b) to compare the new methods with the baseline algorithm for network equilibrium problem, that is, the Frank-Wolfe method;

(c) to evaluate the possible advantages of the inexact decomposition algorithms compared with (approximately) exact decomposition methods and with methods not using a decomposition strategy.

Concerning point (a), we remark that both Algorithm IDA-OD and Algorithm IDA-O require to store path variables that are positive. As shown in Table 7, the number of paths stored at each iteration is very small, and this confirms that the adopted column generation strategy is a viable technique for tackling large dimensional problems.

4.1 Test problems

We performed numerical experiments using the freely available data sets from the repository of Hillel Bar-Gera[1] and listed in Table 1. The arc cost is expressed by the BPR function (see for instance [8, 14]) defined, for each link $i \in \{1, \ldots, m\}$, as

$$s_i(y_i) = \phi_i + \phi_i \beta_i \left(\frac{y_i}{c_i} \right)^{\alpha_i}, \tag{18}$$

[1] `http://www.bgu.ac.il/~bargera/tntp/`

where $C_i, \beta_i, \alpha_i, \phi_i$ are arc dependent parameters, that characterize the traffic network, and

$$y_i = \sum_{j=1}^{n} h_{ij} x_j$$

is the arc flow being h_{ij} elements of the arc-path incidence matrix. Then, according to the adopted notation, we have

$$F_i(y_i) = \int_0^{y_i} \left[\phi_i + \phi_i \beta_i \left(\frac{z}{c_i} \right)^{\alpha_i} \right] dz.$$

name	#nodes	#links	#zones	# OD-pair
Barcelona	1020	2522	110	7922
Winnipeg	1067	2975	154	4344
Berlin Central	12981	28376	865	49688
Chicago Sketch	933	2950	387	93135

Table 1: Data set information.

Performance evaluation

For all tests we report the so-called *relative gap* (see for instance [1] for more details), a widely used quality function defined as

$$r_{gap}(x) = 1 - \frac{\sum_{p=1}^{P} \pi_p(x) d_p}{\sum_{i=1}^{m} s_i(y_i) y_i} = 1 - \frac{\sum_{p=1}^{P} \pi_p(x) d_p}{\sum_{i=1}^{m} s_i(\sum_{j=1}^{n} h_{ij} x_j) \sum_{j=1}^{n} h_{ij} x_j} \tag{19}$$

We recall that π_p is the shortest path cost for the p-th OD pair. The denominator represents the sum, for each arc, of the arc cost weighted by the flow that moves through that arc. It is easy to see that, by the Wardrop equilibrium conditions, $r_{gap}(x^k)$ converges to zero if and only if x^k converges to a solution of the network equilibrium problem.

Implementation details: initial solution, projection over a simplex and scaling

An initial feasible solution, needed to initialize all the implemented algorithms, has been obtained by computing the shortest path tree from each origin (following the origin id order) to all destinations and loading on each OD pair shortest path the entire demand. We have adopted the strategy *once-at-a-time* proposed in [15]. More in particular, the origins are sequentially considered, once that a single origin has been processed, the flow vector and the corresponding path cost vector are reevaluated. In this way, when computing the shortest paths tree for a given origin (with the exception of the first processed origin), we are not considering the graph with an empty flow, so that a better starting point can be obtained at no additional cost. Formally, the vector x is initialized using Algorithm 3, where O is the set of origins.

Algorithm 3: Finds a feasible flow x on the graph G for a given demand d.

Data: A flow-dependent graph G, a set of P OD pairs with corresponding demand vector d.

```
set x = 0;
foreach i ∈ O do
    let t_i be the shortest path tree from i on G;
    foreach p ∈ P with origin i do
        let h_w ∈ t_i be the shortest path between the pair p;
        let d_w the demand of the pair p;
        set x_{h_p} = d_p;
    end
    update arc costs with the current path flow x;
end
```

The projected gradient methods require to project a point over a simplex. Such operation is performed by a very simple algorithm (see [12] for the details), which finds the projection of a point in at most n iterations.

We have also implemented a scaled version of IDA-O algorithm (named

IDA-SO, see later). We recall that in IDA-O algorithm each block component refers to several OD pairs having the same origin. Then, in order to take into account that different OD pairs may have different scales and may generate very different steps, we have scaled the search direction by premultiplying it for a diagonal matrix having on the diagonal, for each OD pair, the maximum steplength (greater than or equal to one) preserving the non-negativity of the corresponding used variables. This should promote a significant updating for all blocks of variables related to different OD pairs. By imposing suitable bounds on the elements of the scaling matrix convergence properties of the algorithm are obviously guaranteed.

4.2 Results

In this section we show the results obtained on the four test problems by the algorithms listed in Table 2. The first three are the gradient projection decomposition algorithms presented in Section 3.1 (the third one is the scaled version), the fourth one is the gradient projection method without decomposition as introduced in Section 3.1. All these algorithms are path-based methods for the solution of formulation (2). The fifth tested algorithm is the Frank-Wolfe algorithm, which is applied to the arc-based formulation (1), and being the most used method, can be considered the baseline algorithm for comparison.

For what concerns the number n_{iter} of projected-gradient iterations for IDA-like algorithms, no significant differences have been observed when performing more than one iteration (say $n_{iter} = 5, 10, 20$). Thus, we only present results in the case of $n_{iter} = 1$.

In order to assess the usefulness of the inexact decomposition strategy characterizing IDA-like algorithms, we have also implemented two corresponding (approximately) exact versions of the same algorithms. More in particular, the

two versions have been realized by setting algorithms IDA-OD and IDA-O n_{iter} sufficiently high (say $n_{iter} = 100$), and by introducing an inner stopping criterion for the solution of each subproblem ($\|d^k\|_\infty \leq 10^{-8}$). These two versions have been named EDA-OD and EDA-O respectively.

algorithm	name
Inexact Decomposition Algorithm-OD	IDA-OD
Inexact Decomposition Algorithm-O	IDA-O
Inexact Decomposition Algorithm-O with Scaling	IDA-SO
Projected Gradient	PG
Frank-Wolfe	FW
Exact Decomposition Algorithm-OD	EDA-OD
Exact Decomposition Algorithm-O	EDA-O

Table 2: Algorithms tested.

All tests have been performed on a Intel Core i7 2.93GHz standard desktop machine with 3GByte of RAM. The algorithms have been coded in C++ using the Boost Graph Library[2] implementation of the Dijkstra algorithm for the shortest path computation, as well as the graph representation.

For each test problem and for each algorithm we report in Tables 3-6 the CPU time required for satisfy the stopping criterion, i.e., for attaining a value of the relative gap (see (19)) less than or equal to the tolerance ϵ, which has been fixed to different values. The symbol $*$ indicates that the algorithm was not able to satisfy the stopping criterion within $5 \cdot 10^3$ seconds for Winnipeg, 10^4 seconds for Barcelona and $2 \cdot 10^4$ seconds for Berlin Central and Chicago Sketch.

From the results reported in Tables 3-6, we can first of all observe that the proposed approach outperforms the standard Frank-Wolfe algorithm both in speed and accuracy in most of the proposed variants. In particular, it can be seen that, despite its ability to quickly reach low accuracy solutions, Frank-Wolfe can not usually make any significant progress towards high quality ones.

[2] `http://www.boost.org/doc/libs/1_46_1/libs/graph/doc/index.html`

algorithm	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-7}$
IDA-OD	11.23	21.43	37.33	109.20
IDA-O	26.93	37.52	60.19	108.96
IDA-SO	1.81	3.46	16.19	40.10
PG	245.45	732.25	818.66	1004.70
EDA-OD	291.81	674.12	805.12	1375.88
EDA-O	107.05	243.54	291.94	411.98
FW	4.09	27.75	*	*

Table 3: Barcelona road network: CPU time (seconds) required to attain $r_{gap}(x^k) \leq \epsilon$.

algorithm	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-7}$
IDA-OD	10.13	23.09	71.60	94.35
IDA-O	14.46	25.74	47.77	88.38
IDA-SO	3.22	18.79	57.11	106.23
PG	64.51	126.09	293.06	469.27
EDA-OD	292.80	755.86	3157.87	4245.42
EDA-O	154.03	478.03	1596.67	2140.79
FW	21.67	147.16	*	*

Table 4: Winnipeg road network: CPU time (seconds) required to attain $r_{gap}(x^k) \leq \epsilon$.

For what concerns the different proposed algorithms that have been tested, we observe that any inexact decomposition-based algorithm outperforms the PG algorithm. Moreover, exact versions are far less efficient than the inexact ones: as conjectured, the effort to exactly optimize a subproblem is not balanced by a substantial improve in the overall convergence speed.

Comparing origin-based and OD-based is tricky: the former is much faster in achieving the same accuracy especially when the network dimension grows. For Barcelona and Winnipeg IDA-O outperforms IDA-OD only when very high accuracy is required; for Berlin Central IDA-OD is better for low and high accuracy, while for the Chicago Sketch network the difference is of one order of magnitude for IDA-OD.

The differences in performance between IDA-O and IDA-OD are mainly due to two factors: the usage of the Dijkstra algorithm to find either single shortest

algorithm	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-7}$
IDA-OD	73.17	167.33	426.13	657.70
IDA-O	39.48	106.27	249.10	809.90
IDA-SO	32.78	106.57	175.14	259.80
PG	154.60	2029.64	10082.73	*
EDA-OD	687.38	1979.10	6809.64	13199.06
EDA-O	397.08	1014.59	2828.35	4444.50
FW	114.96	1360.90	*	*

Table 5: Berlin Central road network: CPU time (seconds) required to attain $r_{gap}(x^k) \leq \epsilon$.

algorithm	$\epsilon = 10^{-4}$	$\epsilon = 10^{-5}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-7}$
IDA-OD	108.07	183.40	261.88	548.67
IDA-O	1095.46	5023.69	8741.1	15640.47
IDA-SO	25.90	384.12	611.04	668.17
PG	9283.45	*	*	*
EDA-OD	2557.88	4142.85	6811.27	14175.96
EDA-O	1076.44	3343.16	8969.27	16363.40
FW	34.72	258.70	*	*

Table 6: Chicago Sketch road network: CPU time (seconds) required to attain $r_{gap}(x^k) \leq \epsilon$.

paths or shortest path trees; the effectiveness of the search directions generated by the projected gradient inner step of the decomposition scheme. For what concern the former, IDA-O can compute shortest path trees from each origin, with a great saving of time. This advantage can be easily verified looking at the time per iteration (i.e. the time to process all OD pairs) which is much lower for IDA-O. On the other hand, search directions generated by IDA-OD seem to be more effective, allowing a greater reduction of the objective function at each iteration.

The previous observations are confirmed by the substantial improvement achieved by the IDA-SO algorithm. The introduction of the scaling matrix seems to substantially improve the practical properties of the scaled descent direction. We observe that in many cases (with different levels of accuracy required) IDA-SO algorithm outperforms all the other tested algorithms. Then,

network	PG	IDA-O	IDA-OD
Barcelona	1.68	1.47	1.49
Winnipeg	1.68	2.08	2.00
Chicago Sketch	2.57	1.38	1.41
Berlin Central	1.13	1.18	1.11

Table 7: Average number of active paths for each OD pair at the best solution found by each algorithm.

on the basis of the experimentation performed on the four test problems, IDA-SO algorithm seems to be the preferable algorithm among those presented in this work.

It is also worth to be noticed that there is room for improvements tuning the gradient projection: the gradient can be scaled by any factor $\lambda > 0$ (which has been set to one in our work) without loosing he convergence properties or any significant computational burden. Scaling the gradient affects the search direction: extensive tests not here reported have shown that tuning λ can improve the algorithms performance.

Being a critical issues for path-based algorithms, we report in Table 7 the average number of active paths for each OD pair recorded at the best solution found by the algorithms IDA-O, IDA-OD and PG.

Values reported in Table 7 clearly show that high quality solutions are characterized by a number of active paths of the same magnitude of the number of OD pairs. Unless the latter became huge, the proposed algorithm is then a viable strategy also in terms of memory consumption.

5 Conclusion

The symmetric network equilibrium problem has been considered in this work. This problem can be cast as a convex optimization problem with a nice structure of the feasible set, i.e. the Cartesian product of simplices.

A column generation strategy has been adopted to overcome the difficulty due to the huge number of variables, so that only active paths are used and stored. This strategy has been combined with a projected gradient method and embedded within an inexact decomposition scheme. The proposed algorithm is then a general framework, named IDA, for which convergence results (under reasonable assumptions) have been given.

From the IDA framework we have derived several specific implementations for the symmetric network equilibrium problem: an OD pairs decomposition, an origin based decomposition and a version with no decomposition at all. For all of them, convergence results have been easily provided exploiting the convergence analysis related to the general decomposition algorithm model.

Computational experiments on standard data sets have shown that the proposed schemes outperform the standard approach based on the well-known Frank-Wolfe algorithm. Results confirm that decomposition is beneficial in any version and that the inexact versions are more efficient than the exact ones.

Memory consumption, one of the main concerns about the usage of the projected gradient algorithms in this context, turns out to be manageable.

A Recalls on Armijo line search

In this section, for sake of completeness and to ease the reader, we recall some known results about Armijo-type line search algorithm.

Let $d^k \in \mathbb{R}^n$ be a feasible direction at $x^k \in \mathcal{F}$. We denote by β^k the maximum feasible step length along d^k.

Assumption 3. *Assume that $\{d^k\}$ is a sequence of feasible search directions such that*

(a) *for all k we have $\|d^k\| \leq M$ for a given number $M > 0$;*

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

An Armijo-type line search algorithm is described below.

Armijo-type line search ALS(x^k, d^k, β^k)

Data: Given $\lambda > 0$, $\delta \in (0,1)$, $\gamma \in (0, 1/2)$ and the initial stepsize $\lambda^k = \min\{\beta^k, \lambda\}$.

Step 1. Set $\alpha = \lambda^k$, $j = 0$.

Step 2. If

$$f(x^k + \alpha d^k) \leq f(x^k) + \gamma\alpha\nabla f(x^k)^T d^k \tag{20}$$

then set $\alpha^k = \alpha$ and stop.

Step 3. Set $\alpha = \delta\alpha$, $j = j + 1$ and go to Step 2.

The properties of Algorithm ALS are reported in the next proposition (see, e.g., [2]).

Proposition 8. *Let* $\{x^k\}$ *be a sequence of points belonging to the feasible set* $\mathcal{F}$*, and let* $\{d^k\}$ *be a sequence of search directions satisfying Assumption* 3*. Then:*

(i) *Algorithm* ALS *determines, in a finite number of iterations, a scalar* α^k *such that condition* (20) *holds, i.e.,*

$$f(x^k + \alpha^k d^k) \leq f(x^k) + \gamma\alpha^k\nabla f(x^k)^T d^k; \tag{21}$$

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

$$\lim_{k\to\infty} \big(f(x^k) - f(x^k + \alpha^k d^k)\big) = 0, \tag{22}$$

then we have

$$\lim_{k\to\infty} \beta^k \nabla f(x^k)^T d^k = 0, \tag{23}$$

where β^k is the maximum feasible step length along d^k.

B Proofs of Propositions 2, 4 and 6

We prove the results stated in propositions 2, 4 and 6 used in the convergence analysis of Algorithm IDA-OD and Algorithm IDA-O of Section 3.1.

Proof of Proposition 2

Proof. **(i)** Let $x \in \mathcal{F}$ and $h \in \{1, \ldots, P\}$. Note that $\mathcal{F}_h(x)$ only depends from $I_h(x)$, and $I_h(x) \subseteq \{1, \ldots, n_h\}$, with $I_h(x) \neq \emptyset$. Then we have

$$|\cup_{x\in\mathcal{F}} \{\mathcal{F}_h(x)\}| = 2^{n_h} - 1$$

(ii) If

$$\bar{x}_{(h)} \in \arg \min_{x_{(h)} \in \mathcal{F}_h(x^k)} f(\bar{x}_{(1)}, \ldots x_{(h)}, \ldots, \bar{x}_{(P)}),$$

then, using the optimality conditions, for any $i \in \{1, \ldots, n_h\}$ such that $\bar{x}_{(h),i} > 0$ we can write

$$\frac{\partial f(\bar{x})}{\partial x_{(h),i}} \leq \frac{\partial f(\bar{x})}{\partial x_{(h),j}} \qquad \forall j \in I_h(x^k). \tag{24}$$

By relabelling, if necessary, the infinite subset K, we have $\pi_h(x^k) = i^\star$ for all $k \in K$, and by definition of $\pi_h(x^k)$, we have

$$\frac{\partial f(x^k)}{\partial x_{(h),i^\star}} \leq \frac{\partial f(x^k)}{\partial x_{(h),j}} \qquad \forall j \in \{1, \ldots, n_h\}. \tag{25}$$

From condition (24), as $i^\star \in I_h(x^k)$, it follows

$$\frac{\partial f(\bar{x})}{\partial x_{(h),i^\star}} \geq \frac{\partial f(\bar{x})}{\partial x_{(h),i}} = \frac{\partial f(\bar{x})}{\partial x_{(h),j}} \qquad \forall i,j \in \{1,\ldots,n_h\} \; s.t. \; x^k_{(h),i},\; x^k_{(h),j} > 0 \quad (26)$$

Now by contradiction assume that

$$\bar{x}_{(h)} \notin \arg\min_{x_{(h)} \in \mathcal{F}_h} f(\bar{x}_{(1)}, \ldots x_{(h)}, \ldots, \bar{x}_{(P)}). \quad (27)$$

Then, there exists a pair $(\hat{i},\hat{j})$ of indexes in $\{1,\ldots,n_h\}$ such that $x_{(h),\hat{j}} > 0$ and

$$\frac{\partial f(\bar{x})}{\partial x_{(h),\hat{i}}} < \frac{\partial f(\bar{x})}{\partial x_{(h),\hat{j}}}. \quad (28)$$

Note that $\bar{x}_{(h),i} > 0$ implies $x^k_{(h),i} > 0$ for $k \in K$ and k sufficiently large. Hence, from (28) and (26), we get that $\bar{x}_{(h),\hat{i}} = 0$ and

$$\frac{\partial f(\bar{x})}{\partial x_{(h),\hat{i}}} < \frac{\partial f(\bar{x})}{\partial x_{(h),j}} \qquad \forall j \in \{1,\ldots,n_h\} \; s.t. \; \bar{x}_{(h),j} > 0. \quad (29)$$

Taking the limits in (25) for $k \in K$ e $k \to \infty$, and recalling (29) we obtain

$$\frac{\partial f(\bar{x})}{\partial x_{(h),i^\star}} \leq \frac{\partial f(\bar{x})}{\partial x_{(h),\hat{i}}} < \frac{\partial f(\bar{x})}{\partial x_{(h),j}} \qquad \forall j \in \{1,\ldots,n_h\} \; s.t. \; \bar{x}_{(h),j} > 0,$$

which contradicts (26). □

Proof of Proposition 4

Proof. **(i)** Similarly to Proposition 2, the set $\Omega_h(x)$ only depends on $I_p(x)$, with $p \in D(h)$. Every $I_p(x)$ is a non-empty subset of $\{1,\ldots,n_p\}$. Then

$$|\cup_{x\in\mathcal{F}} \{\Omega_h(x)\}| = \prod_{p\in D(h)} (2^{n_p} - 1)$$

(ii) By point (ii) of Proposition 2, for every $p \in D(h)$, it holds

$$\bar{x}_{(p)} \in \arg \min_{x_{(p)} \in \mathcal{F}_p} f(\bar{x}_{(1)}, \dots x_{(p)}, \dots, \bar{x}_{(P)})$$

Then, since the set Ω_h is separable, it also holds

$$\bar{x}_{(h)} \in \arg \min_{x_{(h)} \in \Omega_h} f(\bar{x}_{(1)}, \dots x_{(h)}, \dots, \bar{x}_{(R)})$$

□

Proof of Proposition 6

Proof. **(i)** Trivially, in this case

$$|\cup_{x \in \mathcal{F}} \Gamma(x)| = 2^n - 1$$

(ii) Again by point (ii) of Proposition 2, for every $p \in D(h)$, it holds

$$\bar{x}_{(p)} \in \arg \min_{x_{(p)} \in \mathcal{F}_p} f(\bar{x}_{(1)}, \dots x_{(p)}, \dots, \bar{x}_{(P)})$$

Then, since the set Γ is separable, it also holds

$$\bar{x}_{(h)} \in \arg \min_{x_{(h)} \in \Gamma} f(\bar{x}_{(1)}, \dots x_{(h)}, \dots, \bar{x}_{(R)})$$

□

References

[1] H. Bar-Gera. Origin–based algorithms for transportation network modeling. Technical report, National Institute of Statistical Sciences, Research

Triangle Park, NC USA, 1999.

[2] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[3] S. Bonettini. Inexact block coordinate descent methods with application to non-negative matrix factorization. *IMA Journal of Numerical Analysis*, 2011.

[4] R. B. Dial. A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration. *Transportation Research Part B: Methodological*, 40(10):917 – 936, 2006.

[5] M. Florian, I. Constantin, and D. Florian. A new look at projected gradient method for equilibrium assignment. *Journal of the Transportation Research Record Board*, 2090(0361-1981):10–16, 2009.

[6] M. Florian and D. W. Hearn. *Hanbooks in OR & MS*, volume 8, chapter Network Equilibrium Models, pages 485–550. Elsevier Science B.V., Amsterdam, 1995.

[7] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear gauss-seidel method under convex constraints. *Operations Research Letters*, 26(3):127 – 136, 2000.

[8] A. J. Horowitz and United States. Federal Highway Administration. *Delay-volume relations for travel forecasting: based on the 1985 Highway Capacity Manual*. US Dept. of Transportation, Federal Highway Administration, 1991.

[9] R. Jayakrishnan, W. K. Tsai, J. N. Prashker, and S. Rajadhyaksha. A faster path-based algorithm for traffic assignment. *Transportation Research Record*, (1443):75–75, 1994.

[10] C-J Lin. On the convergence of the decomposition method for support vector machines. *Neural Networks, IEEE Transactions on*, 12(6):1288–1298, 2001.

[11] Z. Q. Luo and P. Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72:7–35, 1992. 10.1007/BF00939948.

[12] C. Michelot. A finite algorithm for finding the projection of a point onto the canonical simplex of rn. *Journal of Optimization Theory and Applications*, 50:195–200, 1986. 10.1007/BF00938486.

[13] Y. Nie. A class of bush-based algorithms for the traffic assignment problem. *Transportation Research Part B*, 44:73–89, 2010.

[14] K. R. Overgaard. Urban transportation planning'traffic estimation. *Traffic Quarterly*, 21(2), 1967.

[15] B. Panicucci, M. Pappalardo, and M. Passacantando. A path-based double projection method for solving the asymmetric traffic network equilibrium problem. *Optimization Letters*, 1(2):171–185, 2007.