A Heuristic for the Bilevel Origin–Destination Matrix Estimation Problem

Jan T Lundgren and Anders Peterson

A Heuristic for the Bilevel Origin–Destination Matrix Estimation Problem

Jan T Lundgren[a]* Anders Peterson[a]

janlu@itn.liu.se andpe@itn.liu.se

December 1, 2007

[a] Linköping University, Department of Science and Technology,

SE-601 74 Norrköping, Sweden

* Corresponding author. Tel.: +46 11 363187; fax: +46 11 363270

E-mail address: janlu@itn.liu.se (J. T. Lundgren)

Abstract

In this paper we consider the estimation of an origin–destination (OD) matrix, given a target OD-matrix and traffic counts on a subset of the links in the network. We use a general nonlinear bilevel minimization formulation of the problem, where the lower level problem is to assign a given OD-matrix onto the network according to the user equilibrium principle. After reformulating the problem to a single level problem, the objective function includes implicitly given link flow variables, corresponding to the given OD-matrix. We propose a descent heuristic to solve the problem, which is an adaptation of the well-known projected gradient method. In order to compute a search direction we have to approximate the Jacobian matrix representing the derivatives of the link flows with respect to a change in the OD-flows, and we propose to do this by solving a set of quadratic programs with linear constraints only. If the objective function is differentiable at the current point, the Jacobian is exact and we obtain a gradient. Numerical experiments are presented which indicate that the solution approach can be applied in practice to medium to large size networks.

1 Introduction

The origin–destination (OD) matrix estimation problem is to find an estimate of some travel demand between pairs of zones in a region. OD-matrices are essential inputs in many transportation analysis models and are useful for example when making decisions about how to modify the traffic network, evaluating the accessibility to different commercial areas or when making forecasts of traffic emissions. Many time dependent systems also require a static OD-matrix as a base or default value. It is therefore of great importance to develop accurate models and methods for estimating OD-matrices.

The estimation is made by using available information about the unknown matrix. Usually this information includes traffic count data, i.e., observed traffic flows on a subset of the links in the network, and an initial approximate OD-matrix. This matrix can be an out-dated matrix, for example obtained from surveys or census data and possibly updated by some growth factors to account for changes in the total flow in the network. When making short time OD-matrix estimations, this matrix typically represents the average traffic situation. In this paper the initial matrix can be used as a target matrix, and the estimation problem can be interpreted as a matrix adjustment problem where the initial matrix is updated based on traffic count information.

The general approach adopted in the literature to estimate an OD-matrix is to appropriately select one matrix among all the matrices satisfying the assignment requirements, i.e., all the matrices that reproduce the observed traffic counts when assigned to the network. Depending on how this matrix is chosen (i.e., optimizing criterion) and how the assignment of the matrix on the paths in the network is made, a variety of models can be formulated. In practice, there are always inconsistencies in the observed count information, and there may be no matrix satisfying the observed counts. Therefore most models allow matrices, which do not exactly reproduce the observed counts. For an introduction to the OD-matrix estimation problem, see for example Cascetta and Nguyen (1988), Bell and Iida (1997) and Ortúzar and Willumsen (2002).

In this paper we use a nonlinear bilevel programming formulation of the estimation problem; a formulation that generalizes most models previously presented for this problem. The upper level objective is defined as a weighted measure of deviation from the target matrix and from the observed counts, and the lower level problem is an equilibrium assignment problem ensuring consistency between the OD-matrix and the link flows.

The problem can be reformulated to a single level problem with an implicitly defined objective function, and we suggest to solve this problem using a general descent algorithm which is an adoption of the well-known projected gradient method. The idea is to evaluate the directional derivatives of the objective function at the current equilibrium point. However, since the objective function is not differentiable at all points, we cannot assure that the directional derivatives constitute a gradient. If a strict complementarity condition holds at equilibrium, i.e., if all shortest path routes in each OD-pair have strictly positive flows, we know that we will obtain a gradient. In practice, this condition will probably not hold and we can therefore only expect to compute approximations of the gradient. This turns our approach into a heuristic.

There are many ways to compute an approximation of the gradient and to design a heuristic. The difficulty is to find a good approximation of the Jacobian matrix stating the sensitivity of the link flows with respect to changes in the OD demand. In order to compute an approximate Jacobian matrix we propose to solve a set of quadratic programming problems. We show under which assumptions this solution approach generates the true gradient. We compare our method to previously presented approaches to compute approximate gradients. The initial ideas for our approach was presented in Drissi-Kaïtouni and Lundgren (1992).

The aim of the paper is to present a solution approach that can be a bridge between relatively simple heuristics (e.g. Spiess, 1990) applied to very large problem instances and theoretically convergent algorithms that so far have been applied to very small problem instances only.

In the next section we formulate the general OD-matrix estimation problem using a bilevel structure, and in Section 3 we present our solution strategy. In Section 4 we formulate the quadratic programming problem for finding an approximation of the Jacobian matrix and we also present a method for solving the problem. Finally, in Section 5, we discuss implementation issues and present computational results.

2 Problem formulation

We consider a transportation network where A is the set of links, I the set of OD-pairs and G the set of feasible OD demands. We are interested in finding a feasible vector (OD-matrix) $g \in G$, where $g = \{g_i\}, i \in I$, consists of the demands for all OD-pairs. The assignment of the OD-matrix onto the links in the network is made according to the assignment proportion matrix $P = \{p_{ia}\}, i \in I, a \in A$, where each element in the matrix is defined as the proportion of the OD demand g_i that uses link a. We will use the notation $P = P(g)$ to remark that, in general, these proportions depend of the demand. When assigned to the network, the OD-matrix induces a flow $v = \{v_a\}, a \in A$, on the links in the network. We assume that observed flows, $\tilde{v}_a$, are available for a subset of the links, $a \in \tilde{A} \subseteq A$, and that a target matrix $\hat{g} \in G$ also is available.

The generic OD-matrix estimation problem can now be formulated as

[**P1**]

$$\begin{aligned} \min_{g,v} \; F(g,v) &= \gamma_1 F_1(g,\hat{g}) + \gamma_2 F_2(v,\tilde{v}), \\ \text{s.t.} \; \sum_{i \in I} p_{ia}(g) g_i &= v_a, \;\; \forall \;\; a \in \tilde{A}, \\ g &\in G. \end{aligned} \tag{1}$$

The functions $F_1(g,\hat{g})$ and $F_2(v,\tilde{v})$ represent generalized distance measures between the estimated OD-matrix g and the given target matrix $\hat{g}$, and between the estimated link flows v and the observed link flows $\tilde{v}$, respectively. The functions are assumed to be convex; typically quadratic, entropy or maximum likelihood functions are used, and they can be designed to account for variation in quality of the given data.

The parameters γ_1 and γ_2 reflect the relative belief (or uncertainty) in the information contained in $\hat{g}$ and $\tilde{v}$; problem [**P1**] can therefore be interpreted as a two-objective problem. The two objectives are expressed in F_1 and F_2, and γ_1 and γ_2 are the corresponding weighting factors. In one extreme case, using $\gamma_1 = 0$, the target matrix will have no influence, and in the other case, using $\gamma_2 = 0$, the target matrix will be reproduced and the observed link flows will have no influence.

Feasibility for the OD-matrix is normally defined by non-negativity only, but it is also possible to add some linear constraints, which for example could bound the deviation from the target matrix in some or all OD-pairs.

Previously proposed models for the OD-matrix estimation problem can be classified into two categories, depending on whether we assume the proportion matrix $P(g)$ to be dependent on g or not. If the assignment of the OD-matrix onto the network is independent of the link flows, that is, if we have an uncongested network, $P(g)$ is a constant matrix. The set of equations (1) can then be formulated as

$$\sum_{i \in I} p_{ia} g_i = v_a, \quad \forall \;\; a \in \tilde{A}. \tag{2}$$

The models presented by van Zuylen (1978), van Zuylen and Willumsen (1980), Carey et al (1981), Willumsen (1981), van Zuylen and Branston (1982), Bell (1983), Willumsen (1984), McNeil and Hendrickson (1985), Spiess (1987), Brenninger-Göthe et al (1989), Bierlaire and Toint (1995) and Bianco et al (2001) all belong to this first category. They differ basically in how the functions F_1 and F_2 are defined and motivated. A constant proportion matrix, often referred to as a splitting rate matrix, is also a base in many models for dynamic OD estimation, where g is assumed to be time dependent, see e.g. Ashok (1996), Ashok and Ben-Akiva (2000) and Sherali and Park (2001).

In the models in the second category it is assumed that the network is a congested network and that the OD demand is assigned to the links with respect to the congestion on the links. The proportion matrix is then dependent on g, but the relationship can only be implicitly defined. The feasible set in [**P1**] is defined as all the points (g, v) where v is the link flow solution satisfying an assignment of the corresponding demand $g \in G$. The first model in this class was presented by Nguyen (1977), and extended models were proposed by Jörnsten and Nguyen (1979), Gur et al (1980) suggesting how to obtain unique OD-matrices. Erlander et al (1979) and Fisk and Boyce (1983) have proposed methods based on a combined distribution and assignment model. In all these papers it is assumed that the assignment is made according to the deterministic user equilibrium assumption, and this will be the case also in the present paper.

Fisk (1988) recognized the OD-matrix estimation problem as a bilevel programming problem and used a variational inequality formulation to express the equilibrium conditions, in this way allowing for general link cost functions.

Let $c_a(v), a \in A$ be the link cost functions in the network, let $h_k, k \in K_i, i \in I$ be the flow on path k where K_i refer to the set of all paths in OD-pair i; and define δ_{ak} as one if link a is included in path k, zero otherwise.

Also, assume that all non-negative OD demands are feasible, i.e. let $G = \{g | g_i \geq 0, i \in I\}$.

Problem [**P1**] can now be formulated as a bilevel programming problem as [**P2**]

$$\begin{aligned} \min_g \; F(g, v) &= \gamma_1 F_1(g, \hat{g}) + \gamma_2 F_2(v, \tilde{v}) \\ \text{s.t.} \qquad g &\geq 0 \end{aligned}$$

where v is the optimal solution to the deterministic equilibrium assignment problem (see e.g. Patriksson, 1994):

$$\min_v \; f(v) = \sum_{a \in A} \int_0^{v_a} c_a(s) ds, \tag{3}$$

$$\text{s.t.} \quad \sum_{k \in K_i} h_k = g_i, \; \forall \; i \in I, \tag{4}$$

$$\sum_{i \in I} \sum_{k \in K_i} \delta_{ak} h_k = v_a, \; \forall \; a \in A, \tag{5}$$

$$h_k \geq 0, \; \forall \; k \in K_i, \forall \; i \in I. \tag{6}$$

In the upper level problem of [**P2**] the vector g, i.e., the OD-matrix, defines the decision variables and we want to minimize $F(g, v)$ subject to $g \geq 0$ and a requirement that the link flow v satisfies the equilibrium assignment conditions for the corresponding g. These conditions are obtained by solving the nonlinear lower level problem in which the link (path) flows are the decision variables.

Problem [**P2**] can be interpreted as an instance of a general nonlinear bilevel programming problem. It is well-known that bilevel programming problems are non-convex problems, and by using methods known today we can at the best obtain solutions which are local optima. Since we use a local search strategy, not leaving the initial sub-convex domain, the choice of initial solution of course is essential for which local optima we will obtain. Therefore the OD-matrix estimation problem is often referred to as an adjustment or calibration problem.

Spiess (1990) suggested a heuristic approach to solve problem [**P2**], assuming $\gamma_1 = 0$ and using $\hat{g}$ as the initial solution in the iterative procedure. In this approach an approximate gradient of the objective function with respect to the OD demands is computed, assuming that the proportion matrix $P(g)$ is locally constant. Florian and Chen (1992, 1995) reformulated the bilevel problem into a single level problem using the concept of marginal function. They proposed to use a Gauss-Seidel type heuristic to solve the problem. Independently, both Chen (1994) and Doblas and Benitez (2005) have proposed to solve the problem by an augmented Lagrangean technique. Chen's approach requires that all used paths in each OD-pair are known beforehand, and thus the approach is only applicable to very small problem instances. The technique proposed by Doblas and Benitez is more efficient and minimizes the amount of stored information.

In the methods proposed by Sherali et al (2003) and Nie et al (2005) no assignment matrix P is needed. Instead a set of shortest paths is computed and the corresponding path flows are estimated, as to reproduce the link flow observations as well as possible. The set of shortest paths is iteratively updated with respect to the travel times induced by the flow estimates. Thus, the dependence between route choice and travel demand is assumed to be locally constant. The differences between these two methods lie in the mathematical modelling of the problem and how the shortest paths are computed and updated.

Yang et al (1992), Yang (1995) and Yang and Yagar (1995) all use the bilevel formulation and propose heuristics, which iteratively solves the upper and lower level problem. Information from the lower level problem is transferred by so called influence factors, which can be defined by path proportions or explicit derivatives. The derivatives are computed based on the sensitivity analysis by Tobin and Friesz (1988). Assuming complementary conditions to hold and disregarding any topological dependencies, they get approximate values of the derivatives, which are acceptable for small to medium sized networks. However, for larger networks the topological dependencies are significantly greater.

Maher and Zhang (1999) also have proposed an iterative solution procedure for the bilevel formulation of the problem. In each iteration a new OD-matrix is computed in two steps. In a first step, the assignment proportions P are kept locally constant, leading to a new tentative OD-matrix. This matrix is used to define a search direction, and by assigning it onto the network, the authors get a linear approximation for how the assignment proportions P are affected along the search direction. As a second step, a line search procedure concludes the iteration.

The models proposed by Cascetta and Postorino (2001), Maher et al (2001) and Yang et al (2001) differ from the other models in the sense that they assume a stochastic user equilibrium. An alternative approach is presented by Codina and Barceló (2004) using a subgradient method for nondifferentiable optimization which does not need any sensitivity information.

To summarize, until today the only method applicable for real size networks seems to be the relatively simple heuristic proposed by Spiess, and, possibly, extended with an extra line search, as was suggested by Maher and Zhang. We will present a method which can also take the influence factor into account, and which can solve the OD-matrix estimation problem with reasonable computational effort also for large networks. As we will see, the method of Spiess can be interpreted as a special case of our approach.

3 General solution strategy

Since the link flows are functions of the OD demands, we can formulate problem [**P2**] equivalently in terms of OD demand variables only as a single level problem according to

[**P3**]

$$\min_{g \geq 0} \; F(g) = \gamma_1 F_1(g, \hat{g}) + \gamma_2 F_2(v(g), \tilde{v}),$$

where $v(g)$ is an implicit function of g, whose value is given by the optimal solution of an equilibrium assignment problem for each g. Problem [**P3**] is an unconstrained problem except for the non-negativity constraints on g.

We propose to use a gradient-based descent algorithm to solve the problem. The idea is to evaluate the directional derivatives of the objective function at the current point, and to obtain a descent direction by making a projection on the non-negativity constraints. After a line search, the procedure is repeated until some stopping criterion is satisfied.

Our solution strategy can be outlined as follows:

0. *Initialization*:

   Set g^0 to some nonnegative value and solve an equilibrium assignment problem for the given g^0. Let v^0 be the corresponding equilibrium link flows. Set $l = 0$.

1. *Computation of a descent direction*:

   Find a direction r^l such that $F(g) < F(g^l) + r^l(g - g^l)$ in a neighborhood of g^l $(g \neq g^l)$. If existing, the gradient $r^l = -\nabla_g F(g^l)$ is a valid descent direction.

2. *Computation of a search direction*:

   If required, adjust the descent direction r^l with respect to the feasibility (non-negativity) conditions for the demands.

   Let $\bar{r}_i^l = \begin{cases} r_i^l, & \text{if } g_i^l > 0, \text{ or if } g_i^l = 0 \text{ and } r_i^l > 0, \\ 0, & \text{otherwise,} \end{cases}$

   be the component of the search direction related to OD-pair i, $\quad \forall i \in I$.

3. *Stopping criterion*:

   Interrupt the procedure if some stopping criterion holds, else continue.

4. *Line search*:

   i/ Find a search limit $\alpha_{\max}^l$ such that the new OD demand always will be feasible, i.e. $\alpha_{\max}^l = \min\{+\infty, -g_i^l/\bar{r}_i^l : \bar{r}_i^l < 0, i \in I\}$.

ii/ Find the best step length, i.e. the step length α^l minimizing

$F(g^l + \alpha\bar{r}^l),\ \alpha \in [0, \alpha^l_{\max}]$.

5. *Update*:

   Set $g^{l+1} = g^l + \alpha^l \bar{r}^l$ and let v^{l+1} be the corresponding equilibrium link flows.

   Set $l = l + 1$ and return to step 1.

Under the assumption that the objective function is differentiable, the directional derivatives $\nabla_g F$ always define a gradient vector of the objective function. Then, using the gradient to get a search direction, the method is nothing but an adoption of the well-known projected gradient method; see e.g. Luenberger (1984). However, the objective function in problem [P3] is non-differentiable at certain points and the gradient may not exist. The partial (or directional) derivatives can then be used to represent an approximation of the gradient $\nabla_g F$. In practice we cannot compute the partial derivatives nor perform the line search exactly. We are satisfied as long as the used approximation of the partial derivatives represents a descent direction in which some sufficient decrease of the objective can be obtained by choosing a proper step length. If this is not the case, the partial derivatives are computed more accurately to avoid the algorithm to switch from one sub convex domain to another. If then a descent direction still cannot be found, then, at latest, the algorithm terminates.

Since we use a local search method, the solution found will depend on the starting point in the iterative procedure, and it is often natural to choose the target demand matrix $\hat{g}$ as the initial solution in step 0 of the algorithm.

In step 2 of the algorithm, a search direction is computed from the descent direction through a projection onto the feasible region. Search directions decreasing OD demands already equal to zero are in this way omitted. This simple projection is only one among many alternative ways of performing a projection.

There are many ways to choose a stopping criterion for the algorithm. Possible choices are the norm of the search direction, the (relative) improvement of the objective in the last iteration(s), a maximum number of iterations, a maximum running time or any combination of these criteria.

The difficulty in step 4 lies in the fact that F is not an explicit function of g. In fact, for each tentative step length we have to solve one equilibrium assignment problem. In practice the equilibrium assignment problems can be very large. It is therefore advantageous to utilize the previous equilibrium solution as starting solution in the next iteration and to reoptimize this solution for each new g (each new α). This can be easily done if the equilibrium solution is explicitly expressed in terms of path flows. Of course any method can be used to solve the equilibrium assignment problem, as long as path flows can be computed. The path flows (or path proportions) are in any case required to initialize the approximate solution scheme in the computation of the directional derivatives, as will be shown in Section 4.

Implementation aspects of the algorithm are discussed in more detail in Section 5.

4 Computing the search direction

In this section we show how to compute directional derivatives of $F(g)$ in order to determine an approximate gradient and a good search direction in the descent algorithm.

A gradient of $F(g)$ can be expressed as

$$\begin{aligned}\nabla F(g) &= \gamma_1 \nabla F_1(g) + \gamma_2 \nabla F_2(v(g))\\ &= \gamma_1 \nabla_g F_1(g) + \gamma_2 \nabla_g v(g)\ \nabla_v F_2(v(g)),\end{aligned}$$

where

$$\begin{aligned}\nabla_g F_1(g) &= \left\{\frac{\partial F_1}{\partial g_i}(g)\right\}, i \in I,\\ \nabla_g v(g) &= \left\{\frac{\partial v_a}{\partial g_i}(g)\right\}, a \in A, i \in I,\\ \nabla_v F_2(v(g)) &= \left\{\frac{\partial F_2}{\partial v_a}(v(g))\right\}, a \in \tilde{A}.\end{aligned}$$

The partial derivatives $\{\frac{\partial F_1}{\partial g_i}\}$ and $\{\frac{\partial F_2}{\partial v_a}\}$ are easily computed for suitable choices of distance measures (e.g. quadratic functions or entropy functions) and link cost functions. The difficulty is the computation of an approximate Jacobian $J = \{\frac{\partial v_a}{\partial g_i}\}$.

Next, in Section 4.1, we present two approximations which correspond to previously presented methods in the literature. Then, in Section 4.2, we present our approach where we suggest to formulate one quadratic programming problem for each OD-pair to obtain the directional derivatives. In Section 4.3 we show how to solve the quadratic programs.

4.1 Alternative approximations of the Jacobian

The first approximation of the Jacobian is obtained by assuming the route proportions to be constant in a neighborhood of the present OD-matrix, i.e. by assuming the set of equations (2) to hold in a neighborhood of g^l. Thus, the term $\frac{\partial v_a}{\partial g_i}$ simply reads p_{ia}. We will further refer to this approximation as *locally constant* or **LC** for short. The solution method using the approximation **LC** differs from the method proposed by Spiess (1990) only in the way the step length is determined. Spiess uses the proportional assumption to yield an analytical expression for the optimal step length, if feasible with respect to the non-negativity restrictions on the demands. Our implementation uses an Armijo-type line search, which will be further specified in Section 5.1. We can note that in the **LC** approximation we are dependent on which route flow solution (of the possibly many route flow solutions consistent with the equilibrium link flows) we use in the computation of the Jacobian. The dependence on the route flow solution distinguishes the **LC** approximation from the other two approximations of the Jacobian, which are discussed below.

An alternative approximation can be made using the sensitivity analysis suggested by Tobin and Friesz (1988), which gives the following analytical expression for the Jacobian:

$$J^T = \Delta B^{-1} \Lambda^T D^{-1}, \tag{7}$$

where

$$\begin{aligned} B^{-1} &= \left[\Delta^T S \Delta\right]^{-1} \\ D^{-1} &= \left[\Lambda B^{-1} \Lambda^T\right]^{-1}. \end{aligned}$$

The matrix $\Delta = \{\delta_{ak}\}$ is the link-path incidence matrix, and $\Lambda = \{\lambda_{ik}\}$ is the OD-pair-path incidence matrix (only the used paths are included in this matrix). The matrix $S = \nabla_v c(v(g))$, with elements $s_{ab} = \frac{\partial c_a}{\partial v_b}(v(g))$, is the Jacobian of the link cost functions $c_a(v), a \in A$, with respect to the link flows $v_b, b \in A$, in the current solution $v(g)$.

The use of the sensitivity analysis by Tobin and Friesz for computing a Jacobian of the link flows with respect to the OD-matrix has been criticized (see Bell and Iida (1997), Section 5.4; Patriksson (2004), Section 2.2; and Yang and Huang (2005), Section 4.3). The criticism can be summarized in the two basic assumptions underlying their approach. First, it relies on the strict complementarity condition in order to guarantee a correct computation of the Jacobian. The strict complementarity condition is however a sufficient but not necessary condition for differentiability. It may therefore happen that the approach is unable to compute a gradient even if the objective function $F(g)$ is differentiable. It may also happen that a point is differentiable, even though strict complementarity does not hold.

The second assumption concerns the absence of topological dependencies in the network. If the number of routes in the network is greater than the number of links, the matrix B is singular and thus not invertible. This will certainly be the case for medium to large size networks. Therefore, in general, no analytical solution exists to equations (7).

If the strict complementarity conditions and the topological dependencies are ignored, it is nevertheless possible to utilize the equations (7) to numerically find an approximation of the Jacobian. We will further refer to this approximation as *explicitly derivation* or **ED** for short. Our strategy for evaluating equality (7) will be further specified in Section 4.2 and the implementation issues are discussed in Section 5.1. Our approach corresponds to Yang (1995), differing only in the way the matrix inversions and line searches are made.

Another way to compute an explicit derivation of the Jacobian is proposed by Yang and Huang (2005). Their approach is built on the use of a subset of the equilibrium paths, fulfilling strict complementarity and linear independence. Implemented and verified for a larger network, where ties are to be broken when the subset is defined, this new sensitivity analysis could be an interesting alternative way to compute an explicitly derived Jacobian.

4.2 Second order approximation of the Jacobian

We suggest to use a second order approximation of the Jacobian, and to compute the directional derivatives solving a set of quadratic programming problems, one for each OD-pair $i \in I$. Our approach can be seen as an extension of the linear approximation approach of Spiess (1990). The approach also holds for non-differentiable cases, and we can solve OD-matrix estimation problems with reasonable accuracy and computational effort also for large size networks. It can be shown (Patriksson, 2004) that if the objective function is differentiable at the current point, our approach gives an exact gradient. The initial ideas for our approach was presented in Drissi-Kaïtouni and Lundgren (1992). We will refer to our approach as *implicitly derivation* or **ID** for short.

At the current OD-matrix solution g^l, and for a given OD-pair $\bar{\imath}$, the quadratic programming problem can be formulated as

[**P4**]

$$\min_x \ \tfrac{1}{2} d^T S d$$

$$\begin{aligned} \text{s.t.} \quad \Lambda x &= e_{\bar{\imath}} && (8) \\ d &= \Delta x. && (9) \end{aligned}$$

Vector x is composed of the partial derivatives of the path flows h_k with respect to a unit change in OD demand $g_{\bar{\imath}}$, i.e. $x_k = \frac{\partial h_k}{\partial g_{\bar{\imath}}}(g^l)$. Vector $e_{\bar{\imath}}$ expresses a unit change, i.e. all elements are zeros, except for element $\bar{\imath}$, which is one. Vector d is the partial derivatives of the link flows with respect to a unit change in OD flow $g_{\bar{\imath}}$, i.e., $d_a = \frac{\partial v_a}{\partial g_{\bar{\imath}}}(g^l)$. The vector d^* of optimal values then becomes row $\bar{\imath}$ in the Jacobian matrix, and the elements represent the variation in link flows when one additional unit of flow is sent in OD-pair $g_{\bar{\imath}}$. The elements in the constant matrix S are $s_{ab} = \frac{\partial c_a}{\partial v_b}(g^l)$. If the link cost functions are separable, S is a diagonal matrix with elements $s_a = \frac{\partial c_a}{\partial v_a}(g^l)$, and the objective function can be expressed as

$\frac{1}{2}\sum_{a\in A} s_a d_a^2$.

Problem [**P4**] can be interpreted as the problem of assigning one additional unit of flow in OD-pair $\bar{\imath}$, such that the change of costs on all paths in each OD-pair, given the current equilibrium link flow solution, is equal.

If the objective function $F(g)$ is differentiable at g^l, the optimal solution to problem [**P4**] will satisfy condition (7), given that strict complementarity holds at equilibrium, i.e., given that the derivatives are computed from a non-degenerate path flow solution of the traffic assignment problem. A proof is given by Drissi-Kaïtouni and Lundgren (1992).

Thus, under the assumption of differentiability, the solution to problem [**P4**] must be unique. Therefore the optimality conditions for [**P4**] can be utilized in an implementation of the **ED** method. Let constraint (9) be substituted in the objective function and introduce the Lagrangean multipliers $\omega = \{\omega_i\}, i \in I$, for constraint (8). The Karush-Kuhn-Tucker conditions for optimality for problem [**P4**] can then be stated as:

$$\begin{bmatrix} \Delta^T S \Delta & -\Lambda^T \\ \Lambda & 0 \end{bmatrix} \begin{bmatrix} x \\ \omega \end{bmatrix} = \begin{bmatrix} 0 \\ e_{\bar{\imath}} \end{bmatrix} \tag{10}$$

The set of equations (10) is linear and can be solved by matrix inversion to yield an explicit expression for the approximation of the Jacobian. Thus the solution (x^T, ω^T) to (10) is one way to find an explicit derived Jacobian. In our implementation of **ED**, the system of equations (10) is solved using the well known LR factorization method.

4.3 Solution of the quadratic problems

Problem [**P4**] is a quadratic program with equality constraints and any classical method designed for such problems can be used to solve it. In this section we show how a projected gradient method can be efficiently adapted. Note that one quadratic problem has to be solved for each OD-pair.

Theorem 1 *The projected gradient w of the objective function in* [**P4**] *evaluated at a solution x (or $d = \Delta x$) is*

$$w_k = \sigma_k - \tilde{\sigma}_i \quad k \in K_i,\ i \in I, \tag{11}$$

where σ_k is the cost of path k with respect to link costs $\mu_a = \sum_{b \in A} s_{ab} d_b$, and $\tilde{\sigma}_i$ is the mean of the path costs σ_k for the used paths $k \in K_i$.

Proof: See Appendix.

Note that σ_k and μ_a are defined in terms of marginal costs. In the optimal solution to this quadratic problem we have $w_k = 0, k \in K_i$, which means that σ_k is equal for all $k \in K_i$. The interpretation of this is that, for a given change in OD demand i, the change of cost on all paths within each OD pair should be equal, which means that the equilibrium conditions are still satisfied. Also, note that when the link costs functions are separable, we have $\mu_a = s_a d_a$.

The algorithm for solving problem [**P4**] can now be formulated as follows:

- **Initialization:**

  Set $\begin{aligned} x_k &= p_k, && \forall k \in K_{\bar{\imath}} \\ x_k &= 0, && \forall k \in K_i,\ i \in I, i \neq \bar{\imath} \end{aligned}$

where $p_k,\ k \in K_{\bar{\imath}}$, is the proportion $\frac{h_k}{g_{\bar{\imath}}}$ of the demand in OD-pair $\bar{\imath}$ assigned on path k at the equilibrium solution. This solution is obviously feasible since $\sum_{k \in K_{\bar{\imath}}} p_k = 1$. Then, compute $d_a = \sum_{k \in K_{\bar{\imath}}} x_k \delta_{ak}$.

- **General iteration:**

  Calculate the projected gradient w and define the descent path direction $y = -w$. Perform a line-search in this direction, or equivalently in the link direction $z = -\Delta w$ by solving

  $$\min_{\alpha} \frac{1}{2}(d + \alpha z)^T S (d + \alpha z).$$

  It is easy to verify that the optimal solution can be expressed as $\alpha = -\dfrac{z^T S d}{z^T S z}$. The vectors d and x are then updated to $d + \alpha z$ and $x + \alpha y$, and we check if the given stopping criterion is satisfied.

We have to solve one quadratic problem [**P4**] for each OD-pair to compute the Jacobian, and in each quadratic problem we have to make one projection (11) for each OD-pair to obtain the projected gradient w. For large networks this procedure is computationally impractical. However, if we restrict the set of paths in equation (11) and only consider the paths in the current OD-pair, $k \in K_{\bar{\imath}}$, we will still obtain a projected gradient. The projection problem will then only concern one OD-pair (see Appendix).

The mutual dependencies between different path flows that are considered can be restricted in different ways, depending on the network size and topology. One idea is to restrict it to the subset of paths originating or terminating at the same node. The same type of restriction can of course also be applied in the **ED** method.

There are several ways to define a stopping criterion in the above algorithm. One possibility is to stop the algorithm when the norm of the projected gradient w is smaller than some ϵ. This will produce a very accurate Jacobian if ϵ is a small real number (say 10^{-4}). An alternative is to stop after a given number of general iterations.

For the special case when zero iterations are performed in the algorithm, i.e., when the $\bar{\imath}$th row in the Jacobian matrix is approximated by the initial solution $d = \Delta x$ ($d_a = \sum_{k \in K_{\bar{\imath}}} \delta_{ak} p_k$), our approach becomes, in essence, equivalent to Spiess (1990). Only if the path proportions p_k are locally constant (with respect to a change of the demand), the initial solutions for all $|I|$ quadratic problems will generate the exact Jacobian. Then

$$v_a = \sum_{k \in k_i} \sum_{i \in i} \delta_{ak} h_k = \sum_{k \in k_i} \sum_{i \in i} \delta_{ak} p_k g_{\bar{\imath}}$$

and

$$\frac{\partial v_a}{\partial g_{\bar{\imath}}} = \sum_{k \in k_i} \delta_{ak} p_k.$$

5 Implementation and numerical results

In an implementation of the solution strategy presented in Section 2, we have to specify each of the steps more carefully and decide which accuracy to use. In this section we will first present the various types of specifications needed in order to design an algorithm for practical use, and describe which specifications we have used in our numerical tests. We will then present a set of numerical tests of the methods on three different networks to show that the proposed algorithm can be used in practice to solve medium to large origin–destination matrix estimation problems. The first of these networks is a smaller test network, whereas the other two are greater and model real cities (Chicago and Stockholm, respectively).

Depending on how the specifications and parameter settings on different levels are made we will of course obtain different computational results. It is, however, not the scope of this paper to investigate the "best" design of the algorithm, only to indicate the significance of the improvements that can be obtained by improving the approximation of the Jacobian matrix.

5.1 Implementation specifications

The most important specification in a practical implementation is how the search direction should be computed. This specification includes the accuracy in the computation of the directional derivatives of the objective function and how the projection on the set of feasible OD demands is made. We will focus on the implementation of the **ID** method, i.e. where problem [**P4**] is solved using the quadratic programming approach described in Section 4.2. In the **ED** method problem [**P4**] is instead solved explicitly from the Karush-Kuhn-Tucker conditions, stated in (10). The matrix inversions are then evaluated with an implementation of the well-known LR factorisation technique. The **LC** method simply is the **ID** method, but with zero iterations in the algorithm suggested in Section 4.2. The rows in the Jacobian are hence directly given from the path flow proportions in the current equilibrium solution.

We remind that, except for the step length calculation, the **LC** method is the same method as is proposed by Spiess (1990). Further, the method **ED** coincides with that proposed by Yang (1995), except for the step length and the matrix inversion calculations. For a fair comparison we have preferred a common step length calculation procedure for all methods, over an exact implementation of the different algorithms.

In the implementation of the **ID** method, the number of iterations in the algorithm proposed in Section 4.2 is essential. In our numerical experiences we have observed that the search direction is deflected just very little after the first iteration. It is therefore more efficient to make a line search, take a step, update and evaluate a new search direction, than to make a very accurate calculation.

When solving [**P4**] by methods **ED** and **ID**, it is inefficient to consider the mutual dependency between all paths in the network. We can restrict the consideration of other paths to those belonging to the same OD-pair only (these methods are denoted **EDi** and **IDi**, respectively) or to all used paths belonging to OD-pairs originating at the same node (these methods are denoted **EDo** and **IDo**, respectively). In fact, in our numerical tests, we have only managed to run the version of the **ED** method with the sharpest restriction of the mutual dependencies, i.e. the **EDi** method, in any of the test examples below. For the **ID** method, we have chosen to use the most liberal restriction, which we can compute, which differs between the different test networks.

The specification of search direction also includes a decision on how the projection of the approximate gradient onto the non-negativity constraints of the OD demand should be made. In our numerical tests we have used an ϵ-tolerance for the projection, i.e.,

$$\bar{r}_i^l = \begin{cases} r_i^l & \text{if } g_i^l > \epsilon_1, \text{ or if } g_i^l \leq \epsilon_1 \text{ and } r_i^l > 0, \\ 0, & \text{otherwise,} \end{cases} \quad \forall i \in I,$$

where the value of ϵ_1 depends on the input data for each specific problem. A too small value will give a too restrictive (small) search-limit in step 4 of the algorithm, and the algorithm will converge very slowly.

An accurate line search is computationally very demanding, since traffic equilibrium has to be computed for every tentative step length $\alpha \in [0, \alpha_{\max}^l]$. Instead we have used an Armijo type line search procedure (Armijo, 1966), initiated with the maximum step length $\alpha = \alpha_{\max}^l$, i.e. the largest possible step length leading non-negative demands in all OD pairs. For a given α, the objective function $F(g^l + \alpha\bar{r}^l)$ is evaluated. If the objective function is significantly improved, i.e. if $F(g^l) - F(g^l + \alpha\bar{r}^l) > \epsilon_2$, the line search interrupts and the algorithm continues. Otherwise, the step length obviously was too large, and we evaluate the objective function for $\alpha = \alpha/\Theta$, where $\Theta > 1$ is a parameter, specified beforehand.

If the objective function after a maximum number of such decreases still is larger than the objective function value in the previous iteration, we conclude that the search direction is probably not a descent direction due to the inexact computation of the equilibrium. We then accept a minor increase of the objective function and compute a new search direction from this new point. By this we enable all methods to run equally many iterations. In a practical application, however, we believe that the algorithm should be interrupted before (or, as soon as) a non-decreasing iteration occurs for the first time. Solving the equilibrium problem more accurate will enable more iterations.

For the implementation we must specify the parameter Θ, as well as the maximum number of decreases. These parameter values must be balanced with respect to the accuracy, with which the equilibrium assignment is computed. In our experiments, as a rule of thumb, if $\bar{r}^l$ is a significantly improving search direction, the optimum step length is the maximum step length, $\alpha^l_{\max}$. Thus, if $\alpha^l_{\max}$ does not decrease the objective, this typically indicates that the search direction is not a descent direction. Therefore, we choose a rapid decrease of the objective and in the experiments we have set $\Theta = 10$. For our experiments no more than three consecutive decreases of the step length has shown to be useful.

If a search direction gives a significant decrease of the objective with the initial step length $\alpha^l_{\max}$, it might be interesting to explore the decrease of the objective by allowing an even larger step length. The thereby negative (and thus, infeasible) OD demands could be forced to zero. From a practical point of view, dealing with large problems, this could speed up the improvement at each iteration. However, from a theoretical point of view, such a step length makes the algorithm to shift between different convex subdomains, and no convergence guarantees can be given at all.

For computing the equilibrium assignment of each certain OD-matrix, as a subroutine we use an algorithm based on disaggregated simplicial decomposition (Larsson and Patriksson, 1992), which explicitly uses the path flow variables. This method is also preferable since it has very good reoptimization capabilities. In our numerical experiments we have solved the equilibrium assignment to a relative accuracy of 0.001 – 0.01 in terms of the objective function in (3), which are the default accuracy levels specified together with each of the networks. As mentioned, the computation of the search direction, of course, is dependent of this level of accuracy, and if we want very accurate gradient computation we have to compute very exact equilibrium solutions.

The specification of functions F_1 and F_2, measuring the deviation from the target matrix and traffic counts, respectively, and the values of the weighting parameters γ_1 and γ_2, is of course very important for the final result. In our tests, unless something else is stated, we have chosen to specify the parameters according to $\gamma_1 = \gamma_2 = 1$, i.e. to give the same weight to every deviation. Both deviation functions F_1 and F_2 are implemented as least square functions; the particular form of F is given by

$$F(g) = \gamma_1 \frac{1}{2} \sum_{i \in I} (g_i - \hat{g}_i)^2 + \gamma_2 \frac{1}{2} \sum_{a \in \tilde{A}} (v_a(g) - \tilde{v}_a)^2,$$

where $\tilde{A} \subseteq A$ is the subset of links on which counts have been recorded. The target demand matrix $\hat{g}$ was used to define the initial demand g^0.

5.2 Tests on a small network

In the first set of tests we used a very small, fictive network, which is known from the literature. We used the network presented in Nguyen and Depuis (1984) including 13 nodes, 19 links and 4 OD-pairs, and the set $\tilde{A}$ was defined to include six links. Each equilibrium assignment problem was solved to a very high accuracy (relative gap of 10^{-5}). The traffic counts were randomly generated from a rectangular distribution around the equilibrium assignment of the reference OD-matrix, i.e. $\tilde{v}_a \in [v_a(\hat{g}) \pm \frac{1}{10} v_a(\hat{g})], \forall a \in \tilde{A}$.

This test example was solved by the three methods **LC**, **ED** and **ID**. The **ED** method was run with a restriction to only consider paths in the same OD-pair, i.e. this is the method denoted **EDi**. The **ID** method was run with no restriction. In terms of CPU time this is a rather fair comparison. We also investigated the **ID** method with a restriction to only consider paths originating in the same origin node, i.e. method **IDo**, to see if this significantly affects the behavior. The results are presented in Table 1. We have noted that the objective function value increases in some iterations. We remind that this is due to the inaccuracy in the computation of the equilibrium assignment.

After about 10 iterations the algorithm converged to a stable value for all four test cases, and after 15 iterations the test runs were interrupted since no further improvement of the objective was obtained. The tests indicate that the more accurately the search direction was computed, the better results were yielded. Thus, method **ID** outperforms the other methods. However, the differences are rather small and we need more tests to draw any conclusion.

5.3 Tests on the Chicago network

In the second set of tests we used a network modeling the city of Chicago (Tatineni et al, 1998), which originally was developed by Chicago Area Transportation Study. We have used a cut of the complete network including 304 links, 112 nodes and 1 089 OD-pairs having nonzero demand. The data are based on estimates from 1990 and the traffic situation corresponding to the morning peak period 6.00–9.00 a.m. is used. Fictive traffic counts are generated for 100 links from a randomly disturbed traffic assignment of the target OD-matrix, as in the previous set of tests. Each equilibrium assignment problem was solved to the accuracy corresponding to a relative gap of $3 \cdot 10^{-3}$.

We used the same methods as in the previous tests, i.e. **LC**, **EDi**, **ID** and **IDo**. The numerical results are summarized in Table 2, now recording the best solution found so far at each iteration. We would like to emphasize the behavior in the first iterations as most representative for illustrating the differences between the methods. In the later iterations, we believe that the accuracies, which are used for the step length procedure and for the solution of the equilibrium assignment, are not precise enough with respect to the small changes of the OD-matrix, to enable a fair comparison of the different models.

We have chosen to present the results up to 25 iterations, which we believe are relatively safe, with resepect to the accuracy in the computation of the search direction and the rough determination of the optimum step length. If we continue and perform more iterations, we get unstable results. A very best objective function value of 2168 is achieved after 39, 40 and 26 iterations with the **LC**, **EDi** and **ID** methods, respectively. The **IDo** method shows its best result at the somewhat higher value of 2174 after 33 iterations.

Also in this case we can conclude that a more accurate computation of the search direction leads to a better result. Of course, we obtain these improvements at the expense of increased computational times. For a given number of iterations, the total computational time for methods **EDi** and **IDo** increased 2-4 times. For method **ID** the increase was up to 8 times compared to the most simple case **LC** and in a larger network this might be a too accurate search direction, i.e. require too much CPU time and memory capabilities.

5.4 Tests on the Stockholm network

The third and final set of tests were made on a network from the city of Stockholm, consisting of 964 links, 417 nodes and 1642 OD-pair. For this network, we only tested and compared the **LC** method and the **IDi** method. The latter method being the **ID** method with a restriction of the influence between different path flows to those belonging to the same OD-pair. An attempt to compare the results with the **IDo** method (which has been used for the previous test networks) failed, because of the demanding computations. All other methods also required too much computational time to be used in practice.

We used $\gamma_1 = 0$ in the objective function, which mean that we did not take the distance from the target matrix into consideration. Each equilibrium assignment problem was solved to the accuracy corresponding to a relative gap of 10^{-3}. The computational results are reported in Table 3. As for the Chicago test network, we point out that the most important differences lie in the first iterations. After approximately 25 iterations the changes of the objective are very small in any of the two methods, and merely dependent on the precise appearance of the computed search direction, and the accuracy used in the line search procedure. Therefore, as for the Chicago network, we only report the first 25 iterations in Table 3. The very lowest objective function values, 1205 and 1074, are achieved after some 50 iterations, by the **LC** and the **IDo** methods, respectively.

Most of the computational time (more than 95 percent) is spent in solving the equilibrium assignment subproblems. The overall computational efficiency of the algorithm is consequently very dependent on the efficiency of the assignment algorithm used. Any algorithm generating path-flows can be used in the solution procedure. The path flow solution is used in the solution of the quadratic subproblems generating the Jacobian, and is also used as a starting solution in the reoptimization of the assignment problem given an updated OD-matrix. The computational time for the initial assignment problem corresponds to the total time for solving the next 20 assignment subproblems. After the initial assignment problem the additional computational time is proportional to the number of assignment problems solved. Since only one assignment problem is solved in most of the iterations, independent of the search direction used, the additional computational time is also proportional to the number of iterations performed.

6 Conclusion

The aim of this paper is to show how a bilevel formulation of the origin–destination matrix estimation problem can be solved using a descent algorithm. The proposed solution approach is general in the sense that in a practical application a number of specifications regarding the detailed design has to be made. The emphasis in this paper has been on the computations of search directions, and our tests indicate that more accurately computed search directions (less approximate determined Jacobians) lead to lower objective function values. We recommend the use of the implicitly derived Jacobian, where the consideration of the mutual dependencies is restricted to one OD-pair (previously denoted **IDi**). The major advantage of this method is that it also can be used for large-scale networks. The method outperforms method denoted **LC**, which essentially is the method of Speiss (1990) and used in practice for very large applications, with respect to objective function value. The extra computational efforts required for method **Idi** in comparison to **LC** is marginal.

Our proposed method can be summarized as follows:

0. Solve an equilibrium assignment problem for the initial OD-matrix, resulting in path proportions for each OD-pair.

1. For each OD-pair:

   – compute the change of path cost for all paths in the OD-pair, given a unit change of the OD flow.

    - compute the mean of the change of path costs over all paths and make one adjustment of the path proportions to equilibrate the change in path costs.
    - compute the path proportions for each of the links and let this vector of updated path proportions be a row in the Jacobian matrix.

2. Compute the search direction using the Jacobian matrix.

3. Take a step in the direction to obtain a new OD-matrix, resolve the equilibrium assignment problem and return to step 1.

The outcome of the OD-matrix estimation problem depends very much on the available input data (initial matrix, set of links with traffic count information, cost functions etc.), the quality of this data, the choice of functions F_1 and F_2, and the choice of weighting factors γ_1 and γ_2. Nevertheless, the algorithm used for solving the problem and finding a good OD-matrix given all these data is of great importance. In this paper we have proposed a general solution approach and discussed specifications of this heuristic for solving the bilevel OD-matrix problem. We believe the heuristic can be applied to large-scale networks and contribute to better OD-matrix estimation.

APPENDIX: Proof of Theorem 1

By substituting $d = \Delta x$ in the objective function, problem [**P4**] can be equivalently formulated

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^T \Delta^T S \Delta x \\ \text{s.t.} \quad & \Lambda x = e_{\bar{\imath}}. \end{aligned}$$

The gradient σ of the objective with respect to x is

$$\sigma \; = \; \Delta^T S \Delta x \; = \; \Delta^T S d,$$

where $\sigma_k = \sum_{a \in A} \mu_a \delta_{ak}$ can be interpreted as the cost on path k given the link costs $\mu_a = \sum_{b \in A} s_{ab} d_b, a \in A$. The projection w on the null space of Λ is obtained by solving the projection program

$$\begin{aligned} \min_w \quad & \frac{1}{2}(\sigma - w)^2 \\ \text{s.t.} \quad & \Lambda w = 0, \end{aligned}$$

which can be rewritten algebraically as

$$\begin{aligned} \min_w \quad & \frac{1}{2} \sum_{k \in K_i, i \in I} (\sigma_k - w_k)^2 \\ \text{s.t.} \quad & \sum_{k \in K_i} w_k = 0, \qquad i \in I. \end{aligned}$$

This problem separates into one problem for each OD-pair i. Let ρ be the dual variables corresponding to the constraints. The Kuhn-Tucker conditions corresponding to subproblem i are

$$\begin{aligned} \sigma_k - w_k - \rho_i &= 0, \qquad \forall\, k \in K_i, \\ \sum_{k \in K_i} w_k &= 0. \end{aligned}$$

Summing the first equations over all $k \in K_i$, and using the second equation, we obtain

$$\sum_{k\in K_i} \sigma_k - \sum_{k\in K_i} w_k - \sum_{k\in K_i} \rho_i = \sum_{k\in K_i} \sigma_k - n_i \rho_i = 0,$$

where n_i is the number of (used) paths ($|K_i|$) in OD-pair i. Therefore,

$$\rho_i = \frac{1}{n_i} \sum_{k\in K_i} \sigma_k = \tilde{\sigma}_i,$$

where $\tilde{\sigma}$ is the mean of $\sigma_k, k \in K_i$, and

$$w_k = \sigma_k - \tilde{\sigma}_i, \quad k \in K_i.$$

References

Armijo, L., 1966, Minimization of functions having Lipschits continuous partial derivatives, Pacific Journal of Mathematics 16(1), 1–3.

Ashok, K., 1996, Estimation and prediction of time-dependent origin–destination flows, PhD thesis, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Ashok, K., Ben-Akiva, M.E., 2000, Alternative approaches for real-time estimation and prediction of time-dependent origin–destination flows, Transportation Science 34(1), 21–36.

Bell, M.B.G., 1983, The estimation of an origin–destination matrix from traffic counts, Transportation Science 17(2), 198–217.

Bell, M.G.H., Iida, Y., 1997, Transportation network analysis, Wiley, West Sussex, United Kingdom.

Bianco, L., Confessore, G., Reverberi, P., 2001, A network based model for traffic sensor location with implications on o/d matrix estimation, Transportation Science 35(1), 50–60.

Bierlaire, M., Toint, P.L., 1995, Meuse: An origin–destination matrix estimator that exploits structure, Transportation Research Part B 29(1), 47–60.

Brenninger-Göthe, M., Jörnsten, K.O., Lundgren, J.T., 1989, Estimation of origin–destination matrices from traffic counts using multiobjective programming formulations, Transportation Research Part B 23(4), 257–269.

Carey, M., Hendrickson, C., Siddarthan, K., 1981, A method for direct estimation of origin/destination trip matrices, Transportation Science 15(1), 32–49.

Cascetta, E., Nguyen, S., 1988, A unified framework for estimating or updating origin/destination matrices from traffic counts, Transportation Research Part B 22(6), 437–455.

Cascetta, E., Postorino, M.N., 2001, Fixed point approaches to the estimation of o/d matrices using traffic counts on congested networks, Transportation Science 35(2), 134–147.

Chen, Y., 1994, Bilevel programming problems: analysis, algorithms and applications, PhD thesis, Publication No.984, Centre de Recherche sur les Transports, Université de Montréal, Montreal.

Codina, E., Barceló, J., 2004, Adjustment of O-D trip matrices from observed volumes: An algorithmic approach based on conjugate directions, European Journal of Operational Research 155(3), 535–557.

Drissi-Kaïtouni, O., Lundgren, J.T., 1992, Bilevel origin–destination matrix estimation using a descent approach, Report LiTH-MAT-R-1992-49, Department of Mathematics, Linköping Institute of Technology, Linköping, Sweden.

Doblas, J., Benitez, F.G., 2005, An approach to estimating and updating origin–destination matrices based upon traffic counts preserving the prior structure of a survey matrix, Transportation Research Part B 39(7), 565–591.

Erlander, S., Nguyen, S., Stewart, N., 1979, On the calibration of the combined distribution–assignment model, Transportation Research Part B 13(3), 259–267.

Fisk, C. S., 1988, On combining maximum entropy trip matrix estimation with user optimal assignment, Transportation Research Part B 22(1), 69–73.

Fisk, C. S., Boyce, D.E., 1983, A note on trip matrix estimation from link traffic count data, Transportation Research Part B 17(3), 245–250.

Florian, M., Chen, Y., 1992, A successive linear approximation method for the O-D matrix adjustment problem, Publication No.807, Centre de Recherche sur les Transports, Université de Montréal, Montreal, Canada

Florian, M., Chen, Y., 1995, A coordinate descent method for the bilevel o–d matrix adjustment problem, International Transaction in Operational Research 2(2), 165–179.

Gur, Y., Turnquist, M., Schneiderm, M., Leblanc, L., 1980, Estimation of an origin–destination trip table based on observed link volumes and turning movements, Vol 1. Report RD-80/034, FHWA, U.S. Department of Transportation, Washington D.C.

Jörnsten, K.O., Nguyen, S., 1979, On the estimation of a trip matrix from network Data, Report LiTH-MAT-R-79-36, Linköping Institute of Technology, Linköping, and Publication No.153, Centre de Recherche sur les Transports, Université de Montréal, Montreal.

Larsson, T., Patriksson, M., 1992, Simplicial decomposition with disaggregated representation for the traffic assignment problem, Transportation Science 26(1), 4–17.

Luenberger, D.G., 1984, Linear and Nonlinear Programming, Addison Wesley, Reading, Massachusetts.

Maher, M.J., Zhang, X., 1999, Algorithms for the solution of the congested trip matrix estimation problem. In: Proceedings of the 14th International Symposium on Transportation and Traffic Theory, Jerusalem, Israel, July 20–23, 1999, A. Ceder (ed.), Pergamon, 445–469.

Maher, M.J., Zhang, X., van Vliet, D., 2001, A bi-level programming approach for trip matrix estimation and traffic control problems with stochastic user equilibrium link flows, Transportation Research Part B 35(1), 23–40.

McNeil, S., Hendrickson, C., 1985, A regression formulation of the matrix estimation problem, Transportation Science 19(3), 278–292.

Nguyen, S., 1977, Estimating an OD matrix from network data: A network equilibrium approach, Publication No.60, Centre de Recherche sur les Transports, Université de Montréal, Montreal.

Nguyen, S., Depuis, C., 1984, An efficient method for computing traffic equilibria in networks with asymmetric transportation costs, Transportation Science 18(2), 185–202.

Nie, Y., Zhang, H.M., Recker, W.W., 2005, Inferring origin–destination trip matrices with a decoupled GLS path flow estimator, Transportation Research Part B 39(6), 497–518.

Ortúzar, J. de D., Willumsen, L.G., 2002, Modelling transport, 3rd Edition, Wiley, West Sussex, United Kingdom.

Patriksson, M., 1994, The traffic assignment problem – models and methods, VSP, Utrecht, The Netherlands.

Patriksson, M., 2004, Sensitivity analysis of traffic equilibria, Transportation Science 38(3), 258–281.

Sherali, H.D., Narayanan, A., Sivanandan, R., 2003, Estimation of origin–destination trip-tables based on a partial set of traffic link volumes, Transportation Research Part B 37(9), 815–836.

Sherali, H.D., Park, T., 2001, Estimation of dynamic origin–destination trip tables for a general network, Transportation Research Part B 35(3), 217–235.

Spiess, H., 1987, A maximum likelihood model for estimating origin–destination matrices, Transportation Research Part B 21(5), 395–412.

Spiess, H., 1990, A gradient approach for the O-D matrix adjustment problem, CRT Pub. No 693, Centre de Recherche sur les Transports, Universit de Montral, Montreal, Canada.

Tatineni, M., Edwards, H., Boyce, D., 1998, Comparison of disaggregate simplicial decomposition and Frank-Wolfe algorithms for user-optimal route choice, Transportation Research Record 1617, 157–162.

Tobin, R.L., Friesz, T.L., 1988, Sensitivity analysis for equilibrium network flow, Transportation Science 22(4), 242-250.

Willumsen, L.G., 1981, Simplified transport models based on traffic counts, Transportation 10(3), 257–278.

Willumsen, L.G., 1984, Estimating time-dependent trip matrices from traffic counts. In: Proceedings of the Ninth International Symposium on Transportation and Traffic Theory, J. Volmüller and R. Hamerslag (eds.), VNU Science Press, Utrecht, The Netherlands, 397–411.

Yang, H., 1995, Heuristic algorithms for the bilevel origin–destination matrix estimation problem, Transportation Research Part B 29(4), 231–242.

Yang, H., Huang, H.-J., 2005, Mathematical and economic theory of road pricing, Elsevier, Oxford, United Kingdom

Yang, H., Meng, Q., Bell, M.G.H., 2001, Simultaneous estimation of the origin–destination matrices and travel-cost coefficient for congested networks in a stochastic user equilibrium, Transportation Science 35(2), 107–123.

Yang, H., Sasaki, T., Iida, Y., Asakura, Y., 1992, Estimation of origin–destination matrices from link traffic counts on congested networks, Transportation Research Part B 26(6), 417–434.

Yang, H., Yagar, S., 1995, Traffic assignment and signal control in saturated road networks, Transportation Research Part A 29(2), 125–139.

van Zuylen, H., 1978, The information minimising method: validity and applicability to transportation planning. In: New Developments in Modelling Travel Demand and Urban Systems, G.R.H. Jansen et al (eds.), Saxon, Farnborough, United Kingdom.

van Zuylen, J.H., Willumsen, L.G., 1980, The most likely trip matrix estimated from traffic counts, Transportation Research Part B 14(3), 281–293.

van Zuylen, J.H., Branston, D.M., 1982, Consistent link flow estimation from counts, Transportation Research Part B 16(6), 473–476.

	Search direction			
iter	**LC**	**EDi**	**ID**	**IDo**
0	4915	4915	4915	4915
1	4504	4259	4330	4297
2	4479	4096	4029	4035
3	4219	4074	4081	4077
4	4073	3989	4006	4049
5	4042	4067	4057	4080
6	4063	4003	3985	4038
7	4084	3939	4006	4054
8	4106	3948	3927	3963
9	4130	3951	3943	3966
10	4049	3955	3870	3965
11	3927	3959	3903	3963
12	3923	3932	3903	3962
13	3923	3940	3903	3963
14	3923	3940	3903	3963
15	3923	3940	3903	3963

Table 1: Objective function value vs number of iterations and type of search direction for the Nguyen-Depuis testproblem.

	Search direction			
iter	**LC**	**EDi**	**ID**	**IDo**
0	2234	2234	2234	2234
1	2233	2233	2224	2231
2	2230	2233	2221	2231
3	2227	2230	2220	2229
4	2227	2230	2220	2229
5	2227	2230	2218	2227
⋮				
10	2227	2228	2209	2220
15	2219	2219	2202	2211
20	2214	2207	2184	2197
25	2198	2197	2171	2191

Table 2: Objective function value vs number of iterations and type of search direction for the Chicago test problem.

	Search direction	
iter	**LC**	**IDi**
0	1956	1956
1	1802	1798
2	1690	1701
3	1617	1624
4	1546	1557
5	1478	1507
$\vdots$		
10	1297	1313
15	1237	1211
20	1217	1150
25	1215	1111

Table 3: Objective function value vs number of iterations and type of search direction for the Stockholm test problem.