# A Partial Linearization Method for the Traffic Assignment Problem

by


Torbjörn Larsson
Athanasios Migdalas
Michael Patriksson

Department of Mathematics
Linköping Institute of Technology
S-581 83 Linköping
Sweden



LiTH-MAT-R-89-06

Revised December 1992

**Abstract**

This paper presents a new solution technique for the traffic assignment problem. The approach is based on an iteratively improved nonlinear and separable approximation of the originally nonseparable objective function, and resembles the Frank-Wolfe algorithm in the sense that the subproblem separates with respect to commodities. Since the single-commodity subproblems are strictly convex, the new algorithm will not suffer from the poor convergence behaviour of the Frank-Wolfe algorithm, which is a consequence of the extreme solutions of its linear subproblems. The solution method is outlined along with convergence results, and a dual approach to the solution of the strictly convex subproblems is described. The performance of the algorithm is illustrated with two numerical examples.

*Keywords:* *Traffic Assignment, Convex Multi-Commodity Network Flows, Feasible Direction Methods, Frank-Wolfe Algorithm, Partial Linearization, Lagrangean Duality, Coordinate Ascent*

*AMS 1980 Subject Classification: Primary: 90B10; Secondary: 90C25.*

# 1 Introduction

The traffic assignment problem is used for determining a traffic flow pattern given a set of origin-destination travel demands and flow-dependent link performance functions of a road network. The two conditions of optimal assignment mainly considered were enunciated by Wardrop (1952), although perhaps Pigou (1920) formulated the first notion of traffic equilibrium in his simple description of road traffic behaviour under congestion. The two Wardrop conditions are the *user equilibrium condition*, and the *system optimum condition*, respectively. The principle of user equilibrium corresponds to the individual users choosing routes minimizing their own travel time. In the system optimum case, the routes are imposed on the users by a decision-maker, minimizing the total cost incurred in the system.

First to consider an optimization formulation of the traffic equilibrium problem and to present necessary conditions for the existence and uniqueness of equilibria are Beckmann *et al.* (1956). The optimization formulation given by Beckmann *et al.* exists if the partial derivatives of the link performance functions form a symmetric Jacobian. In the asymmetric case, the equilibrium problem has been reformulated as a variational inequality problem (Smith, 1979 and Dafermos, 1980), and also as a nonlinear complementarity problem (Aashtiani, 1979), a fixed point problem (Netter and Sender, 1970 and Asmuth, 1978), and as nonconvex optimization problems by the use of so called *gap functions* (see e.g. Hearn *et al.*, 1984). Dafermos and Sparrow (1969) show that the user equilibrium problem may be formulated as a convex program, under the assumptions that the link performance functions are monotone and form a symmetric Jacobian. In this paper, we consider the user equilibrium problem with fixed travel demands, and assume that the performance function for a specific link is independent of the flow on other links, thus creating a diagonal Jacobian of the cost functions. The proposed method is, however, also valid for the system optimum case.

The development of formalized methods for the solution of traffic assignment problems arose in the middle of the 1960's. The most influential and well known algorithm is an adaptation of the method of Frank and Wolfe (1956), first suggested for use in this field by Bruynooghe *et al.* (1969) and implemented for a small city by LeBlanc *et al.* (1975). The Frank-Wolfe algorithm will in the subproblem preserve the network structure of the problem, but it is well known that the speed of convergence is far from satisfactory. This emerges from the low quality of the search directions generated by the linear subproblems. To overcome this problem, there have been many suggestions for modifications of the original algorithm; among these are *away steps* (Wolfe, 1970), the accelerated Frank-Wolfe algorithms of Meyer (1974) and *simplicial decomposition* (von Hohenbalken, 1975, 1977). Specialized implementations of these modified algorithms for the case of traffic assignment have also been presented.

Another important class of algorithms employed to traffic assignment is the class of block

Gauss-Seidel, or *cyclic decomposition*, methods (see e.g. Ortega and Rheinboldt, 1970). A cyclic decomposition method is a block-coordinatewise search procedure, where sets of variables are fixed at feasible values. The problem is solved in the non-fixed variables by standard nonlinear techniques, and the optimal values of the non-fixed variables are used in the cyclic block fixing scheme. In the traffic assignment context, the cyclic decomposition is made over flow commodities (see e.g. Nguyen, 1974 and Petersen, 1975). The advantage of these algorithms, as compared to the Frank-Wolfe approach, is that the nonlinearity of the objective is preserved, thus better retaining the properties of the original problem. A disadvantage, however, is that these algorithms are inherently sequential, as opposed to the Frank-Wolfe algorithm.

The algorithm to be presented in this paper is an application of a general solution method for nonlinear programs over Cartesian product sets, proposed by Larsson and Migdalas (1990). It is a generalization of the Frank-Wolfe algorithm that avoids extreme point solutions to the subproblems by retaining them nonlinear. In this way, the subproblems will better resemble the original problem, and therefore hopefully provide better search directions. By making the subproblems separable with respect to the flow commodities, the algorithm will avoid the sequential character of cyclic decomposition schemes, and hence, the algorithm may utilize parallel computing environments. The reader may note that the solution method applies to general convex multi-commodity network flow problems, but we limit ourselves to the problem of traffic assignment for the simplicity of the presentation.

In the next section, the traffic equilibrium problem is formulated. Then in Section 3 the algorithm is presented, with convergence results. A dual approach for the subproblems is given in Section 4, and in Section 5, some computational experience is given. Finally, we draw some conclusions and suggest topics for further research.

## 2 The traffic equilibrium model

Using the notation of Table 1, the problem of determining the assignment according to user equilibrium can be stated as the following convex multi-commodity network flow problem (Nguyen, 1974).

$$
[\mathbf{TAP}] \qquad \begin{aligned}
\min \quad T(\mathbf{f}) &= \sum_{(i,j)\in\mathcal{A}} \int_0^{f_{ij}} t_{ij}(s)ds \\
\text{s.t.} \quad \sum_{j\in\mathcal{W}_i} f_{ij}^k - \sum_{j\in\mathcal{V}_i} f_{ji}^k &= \begin{cases} \sum_{i\in\mathcal{D}_k} r_{ki} & \text{if } i = o_k \\ -r_{ki} & \text{if } i \in \mathcal{D}_k \\ 0 & \text{otherwise} \end{cases} \qquad \forall\, i\in\mathcal{N} \qquad \forall\, k\in\mathcal{C} \\
\sum_{k\in\mathcal{C}} f_{ij}^k &= f_{ij} \qquad \forall\, (i,j)\in\mathcal{A} \\
f_{ij}^k &\geq 0 \qquad \forall\, (i,j)\in\mathcal{A} \quad \forall\, k\in\mathcal{C}
\end{aligned}
$$

| Notation | Definition |
|---|---|
| $\mathcal{N}$ | set of nodes |
| $\mathcal{A}$ | set of links |
| $\mathcal{C}$ | set of commodities; defined by the set of origins |
| $o_k$ | origin of commodity $k$ |
| $\mathcal{D}_k$ | set of destination nodes for commodity $k$ |
| $\mathcal{W}_i$ | subset of nodes being terminal nodes of links initiated at node $i$ |
| $\mathcal{V}_i$ | subset of nodes being initial nodes of links terminating at node $i$ |
| $r_{ki}$ | desired flow for destination $i$ in commodity $k$ |
| $f_{ij}^k$ | flow on link $(i,j)$ for commodity $k$ |
| $f_{ij}$ | total flow on link $(i,j)$; equals $\sum_{k\in\mathcal{C}} f_{ij}^k$ |
| $t_{ij}(f_{ij})$ | link performance function for link $(i,j)$ |

Table 1: Notation

Because of congestion the travel times on streets and intersections are strictly increasing functions of link flows. Several types of *link performance functions* have been suggested in order to capture this relationship. The model used in this paper is a generalization of the commonly used standards of the Bureau of Public Roads (1964)

$$t_{ij}(f_{ij}) = t_{ij}^0 \left[ 1 + p \left( \frac{f_{ij}}{c_{ij}} \right)^{m_{ij}} \right].$$

Here, $t_{ij}^0$ is the *free-flow travel time* on link $(i,j)$, and $c_{ij}$ is its so called *practical capacity*; $p$ is a nonnegative constant, and $m_{ij} \geq 1$ is an integer.

The reader should note that an inherent property of [**TAP**] is that its objective function is nonseparable with respect to commodity flows.

# 3 The partial linearization algorithm

The partial linearization algorithm (Larsson and Migdalas, 1990) will, for simplicity, first be described for a general problem over Cartesian product sets. The problem is stated as

$$[\mathbf{P}] \qquad \begin{aligned} \min \quad T(\mathbf{x}) &= \sum_{j=1}^{n} T_j(\mathbf{x}_j) + g(\mathbf{x}_1, \ldots, \mathbf{x}_n) \\ \text{s.t.} \quad \mathbf{x}_j &\in \mathbf{X}_j, \quad j = 1, \ldots, n, \end{aligned}$$

where $\mathbf{X}_j \subseteq \Re^{p_j}$, $p_j \geq 1$. It is assumed that the feasible set $\prod_{j=1}^n \mathbf{X}_j$ is compact, convex and nonempty, and that the objective function $T(\mathbf{x})$ is continuously differentiable and convex. The functions $T_j(\mathbf{x}_j)$, $j = 1, \ldots, n$, are assumed strictly convex whereas the function $g(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is allowed to be nonconvex.

Given a feasible point $\mathbf{x}^{(l)}$, the first step of the algorithm is to approximate the function $g(\mathbf{x})$ by a first order Taylor expansion at $\mathbf{x}^{(l)}$. This gives rise to the subproblem

$$[\mathbf{SUB}] \qquad \begin{aligned} \min \quad & T^{(l)}(\mathbf{x}) = \sum_{j=1}^{n} T_j(\mathbf{x}_j) + g(\mathbf{x}^{(l)}) + \nabla g(\mathbf{x}^{(l)})^{\mathrm{T}}(\mathbf{x} - \mathbf{x}^{(l)}) \\ \text{s.t.} \quad & \mathbf{x}_j \in \mathbf{X}_j, \quad j = 1, \ldots, n. \end{aligned}$$

Clearly, [**SUB**] is separable into $n$ problems. Letting $\nabla g(\mathbf{x}^{(l)})^{\mathrm{T}} = (\mathbf{c}_1^{\mathrm{T}}, \ldots, \mathbf{c}_n^{\mathrm{T}})$ and dropping the constant term, the $j$:th subproblem can be written as

$$[\mathbf{SUB}_j] \qquad \begin{aligned} \min \quad & T_j(\mathbf{x}_j) + \mathbf{c}_j^{\mathrm{T}} \mathbf{x}_j \\ \text{s.t.} \quad & \mathbf{x}_j \in \mathbf{X}_j. \end{aligned}$$

Let $\overline{\mathbf{x}}^{(l)}$ be the unique solution of [**SUB**] at iteration $l$. Then $\mathbf{d}^{(l)} = \overline{\mathbf{x}}^{(l)} - \mathbf{x}^{(l)}$ is a feasible descent direction for $T(\mathbf{x})$, unless $\mathbf{d}^{(l)} = \mathbf{0}$, in which case $\mathbf{x}^{(l)}$ is an optimal solution to [**P**]. The final step of the algorithm is to determine a maximal steplength with respect to the feasible set and perform a line search along $\mathbf{d}^{(l)}$ to give a new iteration point $\mathbf{x}^{(l+1)}$.

It can be shown that this procedure either terminates in a finite number of iterations, in which case an optimal solution has been found, or it generates an infinite sequence $\left\{\mathbf{x}^{(l)}\right\}_{l=1}^{\infty}$ such that any accumulation point is optimal to [**P**]. For proofs concerning the properties of the algorithm the reader is referred to Larsson and Migdalas (1990). The algorithm is clearly a generalization of the Frank-Wolfe method, in the sense that only a part of the objective function is linearized. Below, we apply the partial linearization algorithm to the program [**TAP**].

In order to apply the algorithm, the objective function $T(\mathbf{f})$ is restated as

$$T(\mathbf{f}) = \sum_{k \in \mathcal{C}} \sum_{(i,j) \in \mathcal{A}} \int_0^{f_{ij}^k} t_{ij}(s)ds + g(\mathbf{f}), \tag{1}$$

where

$$g(\mathbf{f}) = \sum_{(i,j) \in \mathcal{A}} \left\{ \int_0^{f_{ij}} t_{ij}(s)ds - \sum_{k \in \mathcal{C}} \int_0^{f_{ij}^k} t_{ij}(s)ds \right\}. \tag{2}$$

Here, the first term in $T(\mathbf{f})$, as expressed in (1), is a separable approximation of the originally nonseparable objective function, and the second term may be interpreted as an error function. The main idea of our approach is then to use this separable objective function instead of the original one, and to take the error function into account by means of a linear approximation of it, as in the general algorithm outlined earlier.

The linearization of $g(\mathbf{f})$ at a feasible solution gives rise to $|\mathcal{C}|$ problems of the form

$$[\mathbf{SUB}_k] \qquad \begin{aligned} \min \quad & \sum_{(i,j) \in \mathcal{A}} \int_0^{f_{ij}^k} \left( t_{ij}(s) + a_{ij}^k \right) ds \\ \text{s.t.} \quad & \sum_{j \in \mathcal{W}_i} f_{ij}^k - \sum_{j \in \mathcal{V}_i} f_{ji}^k = \begin{cases} \sum_{i \in \mathcal{D}_k} r_{ki} & \text{if } i = o_k \\ -r_{ki} & \text{if } i \in \mathcal{D}_k \\ 0 & \text{otherwise} \end{cases} \qquad \forall\, i \in \mathcal{N} \\ & f_{ij}^k \geq 0 \qquad \forall\, (i,j) \in \mathcal{A}, \end{aligned}$$

where $a_{ij}^k$ is the partial derivative of $g(\mathbf{f})$ with respect to $f_{ij}^k$, evaluated at the linearization point.

The objective function of [$\mathbf{SUB}_k$] has a nice interpretation; it fully captures the interactions between travelers from the same origin, while the interactions between travelers from different origins are approximated only. It is also worth noting that when a solution is approached, the algorithm will in fact produce an iteratively improved separable approximation of the original nonseparable objective function, so that, in the limit, the solution to the $|\mathcal{C}|$ subproblems [$\mathbf{SUB}_k$] also solves [**TAP**]. This is evident from the convergence proof in Larsson and Migdalas (1990), which states that the sequence $\left\{\overline{\mathbf{x}}^{(l)}\right\}_{l=1}^{\infty}$ of subproblem solutions also tends towards an optimal solution to [**P**].

Each subproblem [$\mathbf{SUB}_k$], $k \in \mathcal{C}$, is a strictly convex single-commodity flow program, and can be solved using some primal method, like the network convex simplex method (Kennington and Helgason, 1980) or the network reduced gradient method (Beck *et al.*, 1983). An alternative is to utilize dual techniques, which are here advantageous because of the structure of [$\mathbf{SUB}_k$]. Such techniques for the solution of convex network optimization problems have been thoroughly discussed in several recent papers, e.g. by Zenios and Mulvey (1985/86), Bertsekas and El Baz (1987), Bertsekas *et al.* (1987) and Zenios and Mulvey (1988).

# 4 A dual approach to the subproblem

Because of the separability of the subproblem, each commodity forms an independent problem. In the further discussions in this section only we will, for notational simplicity, therefore drop the index $k$ that denotes the commodity. Hence, $f_{ij}$ will denote the single-commodity flow on the link $(i,j)$, $r_i$ the desired flow to destination $i$, and so on. To even more distinguish the problem of this section from the problem [$\mathbf{SUB}_k$] in the last section, the problem will be referred to as [**PR**].

Introducing Lagrangean multipliers (dual variables) $\boldsymbol{\pi} \in \Re^{|\mathcal{N}|}$, the Lagrangean function of [**PR**] with respect to the network constraints is given by

$$L(\mathbf{f}, \boldsymbol{\pi}) = \sum_{(i,j)\in\mathcal{A}} \int_0^{f_{ij}} (t_{ij}(s) + a_{ij})\, ds + \sum_{(i,j)\in\mathcal{A}} (\pi_i - \pi_j)\, f_{ij} + \sum_{i\in\mathcal{D}} (\pi_i - \pi_o) r_i,$$

where $\pi_o$ is the dual variable corresponding to the origin node. The Lagrangean dual of [**PR**] can then be written as

[**PR-LD**] $\quad \max\ \theta(\boldsymbol{\pi})$, where

$$\theta(\boldsymbol{\pi}) = \sum_{i \in \mathcal{D}} (\pi_i - \pi_o) r_i + \sum_{(i,j) \in \mathcal{A}} \min_{f_{ij} \geq 0} \int_0^{f_{ij}} (t_{ij}(s) + a_{ij} + \pi_i - \pi_j)\, ds.$$

Here, $\theta(\boldsymbol{\pi})$ is finite, concave and differentiable (e.g. Bazaraa and Shetty, 1979, Theorems 6.3.1 and 6.3.3). The unique minimizer for each of the $|\mathcal{A}|$ single-variable Lagrangean subproblems can be explicitly expressed as

$$f_{ij}(\pi_j - \pi_i) = \begin{cases} t_{ij}^{-1}(\pi_j - \pi_i - a_{ij}) & \text{if } t_{ij}^{-1}(\pi_j - \pi_i - a_{ij}) \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $t_{ij}^{-1}(\cdot)$ is the inverse function of $t_{ij}(\cdot)$. (Note that the arc flows are nondecreasing functions of the differences $\Delta\pi_{ij} = \pi_j - \pi_i$.)

The problem [**PR**] always has a bounded solution and the below standard results follow (see e.g. Bazaraa and Shetty, 1979, Theorems 6.2.4 and 6.5.2). Here, $v[\cdot]$ denotes the optimal value of a program.

1. $v$[**PR-LD**] $= v$[**PR**]
2. If $\boldsymbol{\pi}^*$ solves [**PR-LD**] then $\mathbf{f}^* = \arg\min_{\mathbf{f} \geq \mathbf{0}} L(\mathbf{f}, \boldsymbol{\pi}^*)$ solves [**PR**].

These results imply the possibility of solving the primal problem [**PR**] implicitly by instead solving the dual problem [**PR-LD**]. How the network structure may be utilized in the solution of [**PR-LD**] will be described in more detail.

As a direct consequence of the strict convexity of [**PR-LD**] the gradient $\nabla\theta(\boldsymbol{\pi})$ is everywhere continuous and its $i$:th element is given by

$$\nabla\theta(\boldsymbol{\pi})_i = \sum_{j \in \mathcal{W}_i} f_{ij}(\pi_j - \pi_i) - \sum_{j \in \mathcal{V}_i} f_{ji}(\pi_i - \pi_j) - r, \text{ where } r = \begin{cases} \sum_{i \in \mathcal{D}} r_i & \text{if } i = o \\ -r_i & \text{if } i \in \mathcal{D} \\ 0 & \text{otherwise.} \end{cases}$$

Algorithms for unconstrained nonlinear programming can thus be adopted to solve the Lagrangean dual [**PR-LD**]. Equivalently, the equation system

[**S**] $$\nabla\theta(\boldsymbol{\pi}) = \mathbf{0}$$

can be solved. Because of the linear dependence of the constraints of [**PR**], the solution is never unique; however, the optimal differences $\pi_j - \pi_i$ will be unique, due to the strictly increasing property of $t_{ij}(f_{ij})$.

We have applied a coordinatewise ascent method to maximize the dual objective; this corresponds to solving [**S**] by satisfying one equation at a time by an update of the corresponding dual variable (e.g. Bregman, 1967). Bregman's method, closely related to the Gauss-Seidel method (e.g. Ortega and Rheinboldt, 1970), has for a long period of time been used in the solution of flow distribution problems (Bregman, 1967 and Lamond and Stewart, 1981) and image reconstruction problems (e.g. Censor, 1981). In terms of network flows in our application, Bregman's method corresponds to balancing the flow through one node at a time.

Let the arc flows be defined according to (3) for $(i,j) \in \mathcal{A}$. Then $\nabla\theta(\boldsymbol{\pi})_i$ is said to be the *divergence* of node $i$. If $\nabla\theta(\boldsymbol{\pi})_i > 0$, the flow leaving node $i$ is strictly greater than the flow arriving at it, and node $i$ is then called a *deficit* node. If $\nabla\theta(\boldsymbol{\pi})_i < 0$, then the situation is the opposite, thus making node $i$ a *surplus* node. If $\nabla\theta(\boldsymbol{\pi})_i = 0$, we say that node $i$ is *balanced*. A deficit node is balanced by increasing the corresponding dual variable, and for a surplus node the dual variable is instead decreased in order to obtain balance. Given some vector $\boldsymbol{\pi}^{(l)}$, the procedure continues by selecting an unbalanced node $i_l$ and balancing it by changing the corresponding dual variable. The balancing is performed in two stages; first a search interval is identified and the new dual variable value is then obtained through a line search within the interval. A new vector $\boldsymbol{\pi}^{(l+1)}$ is obtained and everything is repeated iteratively.

First, we will present the algorithmic concept. Then possible implementations are discussed and justified.

Input is the network $(\mathcal{N}, \mathcal{A})$ with flow demands and the dual function $\theta(\boldsymbol{\pi})$. Output is a vector of approximately optimal dual variables corresponding to an $\varepsilon$-feasible approximation of the optimum of [**PR**].

## Flow balancing algorithm

- **Initialization:**

  Let $\boldsymbol{\pi}^{(0)}$ be any vector of $|\mathcal{N}|$ elements; $\varepsilon > 0$; $l = 0$ and let $\omega \in (0,2)$.

- **Node selection and balancing:**

  Choose $i_l \in \mathcal{N}$ such that $\left|\nabla\theta(\boldsymbol{\pi}^{(l)})_{i_l}\right| > \varepsilon$.
  If $\nabla\theta(\boldsymbol{\pi}^{(l)})_{i_l} > \varepsilon$ then find $\pi_{i_l}^{\max}$ defining the search interval, and balance equation $i_l$ by increasing $\pi_{i_l}$.
  If $\nabla\theta(\boldsymbol{\pi}^{(l)})_{i_l} < -\varepsilon$ then find $\pi_{i_l}^{\min}$ defining the search interval , and balance equation $i_l$ by decreasing $\pi_{i_l}$.
  The solution is $\delta^{(l)}$.

- **Dual update:**

Let

$$\pi_{i_l}^{(l+1)} = \pi_{i_l}^{(l)} + \omega(\delta^{(l)} - \pi_{i_l}^{(l)}), \tag{4}$$

and for all $i \in \mathcal{N},\ i \neq i_l$, let $\pi_i^{(l+1)} = \pi_i^{(l)}$.

- **Convergence check:**

  If $\left|\nabla\theta(\boldsymbol{\pi}^{(l+1)})_i\right| \leq \varepsilon$ for all $i \in \mathcal{N}$, then terminate with $\boldsymbol{\pi}^{(l+1)}$ being an approximate solution.

  Otherwise, let $l := l+1$ and go to the node selection and balancing phase.

The steps of the algorithm can be carried out in several different ways, which are outlined below.

Let $\{i_l\}_{l=0}^{\infty}$ be the sequence of nodes as they are visited. We have considered two ordering strategies: cyclic order and most unbalanced node (or Gauss-Southwell) order. The latter criterion selects the steepest coordinate ascent direction. There are of course other possible orderings, among those threshold order and forward/backward order (e.g. Luenberger, 1984). A survey of orderings is found in Censor (1981).

The value of the potential of node $i$, here called $\delta^*$, balancing the node, must be such that for fixed $\boldsymbol{\pi}$,

$$[\mathbf{E}_i] \qquad \theta_i'(\delta^*) \overset{\text{def}}{=} \nabla\theta(\boldsymbol{\pi} + (\delta^* - \pi_i)\mathbf{e}_i)_i = 0,$$

where $\mathbf{e}_i$ is the $i$:th unit vector. Solving the equation $[\mathbf{E}_i]$ is equivalent to perform a line search along $\mathbf{e}_i$, i.e.,

$$[\mathbf{D}_i] \qquad \max_{\delta} \theta_i(\delta) \overset{\text{def}}{=} \theta(\boldsymbol{\pi} + (\delta - \pi_i)\mathbf{e}_i).$$

The concavity of $\theta(\boldsymbol{\pi})$ implies the concavity of $\theta_i(\boldsymbol{\pi})$. Therefore $\theta_i'(\delta)$ is decreasing with respect to $\delta$ (see Bazaraa and Shetty, 1979, Theorem 6.3.1).

The solution $\delta^*$, to $[\mathbf{E}_i]$, is not surely unique, since $\theta_i'(\delta)$ is not strictly decreasing. Now, suppose that we have selected a node $i$ for which $|\nabla\theta(\boldsymbol{\pi})_i| > \varepsilon$. The algorithm proceeds by determining a point $\hat{\delta}$ equal to $\pi_i^{\max}$ or $\pi_i^{\min}$, such that $\nabla\theta(\boldsymbol{\pi})_i$ and $\theta_i'(\hat{\delta})$ have different signs. The points $\pi_i$ and $\hat{\delta}$ are used to bracket the range of values of $\delta$ where $[\mathbf{E}_i]$ or $[\mathbf{D}_i]$ is solved, using some technique for one-dimensional search. Because of the non-negativity restrictions on the primal variables, $\nabla\theta(\boldsymbol{\pi})_i$ is not everywhere differentiable. It is therefore natural to utilize line-search algorithms requiring only objective function evaluation and first order information.

The bracket is constructed using a simple algorithm, where inputs are $\pi_i$, the sign of $\nabla\theta(\boldsymbol{\pi})_i$, and *a priori* set parameters $\alpha > 0$ and $h > 0$. The parameter $\alpha$ is used to indicate how much the length of the interval is to be increased in each step of the algorithm; often used is $\alpha = 2$. Initially $\hat{\delta}_i = \pi_i$ and $\overline{\delta}_i = \pi_i$ is set. According to the sign of $\nabla\theta(\boldsymbol{\pi})_i$, the search direction is determined. Assume that $\nabla\theta(\boldsymbol{\pi})_i > 0$. If $\theta_i'(\hat{\delta} + h)$ and $\theta_i(\overline{\delta})$ have different signs we terminate with $\hat{\delta} := \hat{\delta} + h$ and $\overline{\delta}$ as end points of the interval. If not, $\overline{\delta} := \hat{\delta}$, $\hat{\delta} := \hat{\delta} + h$ and $h := \alpha h$ is set and the algorithm proceeds iteratively. The algorithm performs analogously for a negative sign of $\nabla\theta(\boldsymbol{\pi})_i$. Important to note here is that the step $h$ may be a constant in every main iteration of the algorithm of partial linearization, but as the iterations proceeds it is appropriate to replace it with a sequence $\{h_l\}_{l=1}^{\infty}$ such that $\lim_{l\to\infty} h_l = 0$.

Assume now that $\delta^{(l)}$ is provided. The value of $\pi_i$ is updated according to the formula (4). In general, three choices of the value of $\omega$ are considered (see e.g. Ortega and Rheinboldt, 1970): projection ($\omega = 1$), underrelaxation ($0 < \omega < 1$), and overrelaxation ($1 < \omega < 2$). For the dual flow balancing algorithm the following convergence theorem is valid.

**Theorem 1** *Let $\left\{\boldsymbol{\pi}^{(l)}\right\}_{l=1}^{\infty}$ be the sequence of dual points generated by the algorithm, using the most unbalanced node ordering and $\omega = 1$ as relaxation parameter. Then the corresponding sequence of primal solutions converges to the optimal solution $\mathbf{f}^*$ of* [**PR**].

**Proof** To prove the theorem, we will apply a result by Bertsekas *et al.* (1987). First, let us assume that all nodes $\pi_i$, $i \in \mathcal{N}$, are selected by the algorithm an infinite number of times. The problem [**PR**] is strictly convex as shown above. Using $\omega = 1$ as steplength in the updating formula (4), i.e. exact projection, our procedure qualifies as a relaxation method according to Bertsekas *et al.* Using Proposition 2.4 in the same reference, the result follows.

Now, suppose that a subset of the nodes of the network, $\tilde{\mathcal{N}}$, is visited only in a finite number of iterations. Then, after some iteration $L$, no node in $\tilde{\mathcal{N}}$ is selected by the algorithm. The complementary set of nodes, $\mathcal{N} \setminus \tilde{\mathcal{N}}$, is selected an infinite number of times, so that the conclusion above holds for this subset. It then follows that the whole set of nodes must be balanced, because otherwise, using the most unbalanced node ordering, one of the nodes in $\tilde{\mathcal{N}}$ would be selected again. This contradicts the assumption that the nodes of $\tilde{\mathcal{N}}$ ceased to be selected after iteration $L$. This completes the proof. $\square$

The update of $\pi_i$ can be performed using other formulas than (4). Because the flow balancing algorithm will be used many times while solving [**TAP**], it is of great interest to minimize the cost of satisfying [**S**]. An algorithm not using an exact flow balancing has therefore also been considered. One example of such updating formulas is

[**M**] $$\pi_i^{(l+1)} = \frac{1}{2}\left(\overline{\delta} + \hat{\delta}\right).$$

Numerical tests actually indicate that the formula [**M**] has a practical advantage over (4), both with respect to time requirements and the number of iterations. To explain this intuitively, the formula [**M**] can be expected to give a stochastic distribution of over- and underrelaxation, respectively, which improves the rate of convergence. It is also obvious that a very accurate balancing of a node is of little advantage, since its balance will be destroyed in a later iteration when the potential of an adjacent node is adjusted.

It should be noted that it is, in practice, impossible to obtain an optimal dual solution $\boldsymbol{\pi}^*$ in a finite number of iterations by the flow balancing algorithm. Accordingly, in our presentation of the dual flow balancing algorithm we only looked for some $\overline{\boldsymbol{\pi}}$-values satisfying $|\nabla\theta(\overline{\boldsymbol{\pi}})_i| \leq \epsilon$, $\forall\ i \in \mathcal{N}$. Using Everett's main theorem (Everett, 1963) it can be shown that a near optimal dual solution $\overline{\boldsymbol{\pi}}$ satisfying the above inequalities corresponds to a near feasible solution, $\overline{\mathbf{f}}$, which is an optimal solution to a primal problem where the right hand side vector of the flow conservation constraints have been perturbated by $\nabla\theta(\overline{\boldsymbol{\pi}})$ exactly. This indicates that the dual approach may be acceptable in practice. Otherwise, one can invoke a heuristic for generating primal feasible solutions within the dual scheme, see e.g. Curet (1992).

# 5 Computational experience

In order to illustrate the performance of the proposed algorithmic concept, two test problems have been solved, one small and one medium scale problem. The method of partial linearization was coded in FORTRAN-77 on a SUN 4/390 computer, using the suggested dual technique for the solution of the subproblems. For both problems, we compare the performance of the proposed method with that of the Frank-Wolfe algorithm. Both algorithms were initiated by the flow given by an all-or-nothing assignment at the zero flow. The dual solution technique was coded using the most unbalanced node selection and updating formula [**M**]. (We have also tested cyclic node selection and exact solution of [$\mathbf{E}_i$] combined with under- or overrelaxation, respectively, but the above choices seem superior.)

The small test example, given in Nguyen and Dupuis (1984), is a quadratic problem of 13 nodes, 19 arcs and 4 origin-destination pairs. Table 2 shows the convergence of the partial linearization versus the Frank-Wolfe method, and two other algorithms; these are the so called *restricted simplicial decomposition algorithm* of Hearn *et al.* (1987), and a cutting plane approach, presented by Nguyen and Dupuis (1984). The results in the table are taken from Nguyen and Dupuis (Frank-Wolfe, Cutting Plane) and from a private communication with S. Lawphongpanich (Simplicial Decomposition). The results from the simplicial decomposition algorithm are worth noting. In the original reference (Lawphongpanich and Hearn, 1984), the algorithm was implemented in single precision, causing the algorithm to generate infeasible points. The figures shown in the table below are obtained by double precision arithmetic. The other results shown in the table are all obtained from single precision arithmetic. After six iterations of the partial linearization method, the deviation from optimality was $4.5 \cdot 10^{-5}\%$, while the Frank-Wolfe method

| Iter. | P.L. | F.-W. | C.P. | S.D. |
|---|---|---|---|---|
| 0 | 125 500.00 | 125 500.00 | 125 500.00 | 125 500.00 |
| 1 | 87 162.65 | 94 253.37 | 94 253.37 | 94 253.38 |
| 2 | 85 192.95 | 86 847.30 | 86 847.30 | 86 560.91 |
| 3 | 85 035.47 | 86 447.56 | 86 294.06 | 85 895.14 |
| 4 | 85 028.98 | 86 229.62 | 86 082.69 | 85 513.68 |
| 5 | 85 028.20 | 85 939.96 | 85 963.00 | 85 215.00 |
| 6 | 85 028.11 | | | 85 047.23 |
| 7 | | | | 85 028.07 |
| 8 | | | | 85 028.07 |
| 10 | | 85 555.79 | 85 210.60 | |
| 15 | | 85 392.72 | 85 061.23 | |
| 20 | | 85 329.47 | 85 035.67 | |
| 25 | | 85 273.72 | 85 030.39 | |
| 30 | | 85 241.00 | 85 028.53 | |
| 35 | | 85 212.33 | 85 028.14 | |
| 40 | | 85 102.52 | | |

Table 2: Comparisons between various methods for the Nguyen/Dupuis problem

reached $8.8 \cdot 10^{-2}\%$ after 40 iterations.

The second test example is a randomly generated grid network of 64 nodes, 112 arcs and 8 origin-destination pairs. The link performance functions used were of the form described in Section 2. In Table 3 we show the performance of the proposed method, and that of the Frank-Wolfe algorithm. The partial linearization method was terminated after 10 iterations, giving a relative accuracy of 0.016%, whereas the Frank-Wolfe method reached a relative accuracy of 0.13% after 50 iterations.

# 6 Conclusions and further research

The proposed feasible direction method acts as a decomposition scheme on partially separable problems, and resembles the Frank-Wolfe algorithm while at the same time differing from it in retaining nonlinear subproblems. The numerical examples illustrate that the new algorithm can be expected to perform considerably better than the Frank-Wolfe algorithm with respect to the number of main iterations. Because of the more information kept in the subproblems, this was expected. The better convergence of the proposed algorithm compared to the Frank-Wolfe method is paid by an increased computational effort required for the subproblem solution. This effort can, however, be reduced by using the optimal dual solution from the previous main iteration as starting solution, since the successive subproblems will differ only slightly in later iterations.

| Iter. | P.L. | F.-W. |
|---|---|---|
| 0 | 81 070.00 | 81 070.00 |
| 1 | 37 076.17 | 46 845.00 |
| 2 | 30 783.53 | 40 126.25 |
| 3 | 29 975.64 | 36 014.45 |
| 4 | 29 555.77 | 32 660.24 |
| 5 | 29 499.12 | 31 538.79 |
| 6 | 29 474.68 | 30 830.22 |
| 7 | 29 469.55 | 30 452.90 |
| 8 | 29 465.46 | 30 155.26 |
| 9 | 29 463.55 | 29 896.63 |
| 10 | 29 462.90 | 29 814.75 |
| 20 | | 29 533.08 |
| 30 | | 29 505.26 |
| 40 | | 29 498.32 |
| 50 | | 29 495.36 |

Table 3: Comparisons between partial linearization and Frank-Wolfe for the grid network

A nice property of the Frank-Wolfe algorithm is that, for convex problems, a termination criterion is available through the lower bound, on the optimal objective value, obtained from the linear subproblem. In the partial linearization algorithm, this termination criterion is, in general, not valid since the function $g(\mathbf{x})$ is not surely convex. It is, however, easy to show that $T\left(\mathbf{x}^{(l)}\right) - T^{(l)}\left(\overline{\mathbf{x}}^{(l)}\right) > 0$ whenever $\mathbf{x}^{(l)}$ is not a solution to [**P**], and that this difference tends towards zero when the iterations proceed. Hence, the difference between the objective value and the corresponding optimal subproblem value may still be utilized as a termination criterion.

During the computational experimentation with our algorithm, we have observed that the dual problems are sometimes rather badly conditioned, leading to numerical difficulties in the flow balancing method. This happens when the partial linearization is made at a point where some arc flows are small while others are large, so that the second order derivatives of the subproblem objective are of different magnitudes. As a result, the corresponding Lagrangean dual problem becomes ill-conditioned; a similar observation is made by Bertsekas *et al.* (1987, p. 1241). A possible counter-measure is to consider the use of primal methods for solving the subproblems. One such alternative is to solve the subproblems approximately by the utilization of a truncated Frank-Wolfe method, that is to make a few Frank-Wolfe iterations only. The application of truncated primal algorithms will be further studied in the near future.

An interesting observation, and a possible subject for further research, is that the given restatement of the original objective function $T(\mathbf{x})$ is by no means the only possible. One can in fact rewrite $T(\mathbf{x})$ as

$$T(\mathbf{x}) = h(\mathbf{x}) + (T(\mathbf{x}) - h(\mathbf{x})),$$

where $h(\mathbf{x})$ is an arbitrary separable strictly convex function, and still use the same algorithmic framework by making linear approximations of the term $T(\mathbf{x}) - h(\mathbf{x})$. As an example, we consider the choice of $h(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{G}\mathbf{x}$, where $\mathbf{G} = \operatorname{diag}\left(\nabla^2 T\left(\mathbf{x}^{(0)}\right)\right)$, which leads to the separable subproblem

$$\min_{\mathbf{x}_j \in \mathbf{X}_j, \forall j} T^{(l)}(\mathbf{x}) = T\left(\mathbf{x}^{(l)}\right) + \nabla T\left(\mathbf{x}^{(l)}\right)^{\mathrm{T}}\left(\mathbf{x} - \mathbf{x}^{(l)}\right) + \frac{1}{2}\left(\mathbf{x} - \mathbf{x}^{(l)}\right)^{\mathrm{T}}\mathbf{G}\left(\mathbf{x} - \mathbf{x}^{(l)}\right).$$

Hence, an approximate Newton algorithm is defined. Through various choices of $h(\mathbf{x})$ the complexity of the subproblem spans from the difficulty of the linear Frank-Wolfe subproblem (by the choice of $h(\mathbf{x}) = 0$), to a subproblem, as difficult to solve as the original problem (by the choice of $h(\mathbf{x}) = T(\mathbf{x})$).

A further generalization of the partial linearization principle is obtained by defining a sequence of function $h^{(l)}(\mathbf{x})$, thus introducing an iteration dependency in the above approximation concept; such schemes are studied in Patriksson (1991a, 1991b). The application of these schemes to traffic assignment problems is a subject for future research.

## Acknowledgements

The research leading to this paper was supported by the Swedish Transport Research Board (TFB) and the Swedish National Board for Technical Development (STU).

# References

[1] Aashtiani, H.Z. (1979), *The multi-modal traffic assignment problem.* Ph.D. dissertation, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA.

[2] Asmuth, R. (1978), *Traffic network equilibrium.* Technical Report SOL-78-2, Stanford University, Stanford, CA.

[3] Bazaraa, M.S. and Shetty, C.M. (1979). Nonlinear Programming. Theory and Algorithms. John Wiley & Sons, New York, NY.

[4] Beck, P., Lasdon, L. and Engqvist, M. (1983), *A reduced gradient algorithm for nonlinear network problems.* ACM Transactions on Mathematical Software **9**, pp. 57–70.

[5] Beckmann, M.J., McGuire, C.B. and Winsten, C.B. (1956). Studies in the Economics of Transportation. Yale University Press, New Haven, CT.

[6] Bertsekas, D.P. and El Baz, D. (1987), *Distributed asynchronous relaxation methods for convex network flow problems.* SIAM Journal on Control and Optimization **25**, 74–85.

[7] Bertsekas, D.P., Hosein, P.A. and Tseng, P. (1987), *Relaxation methods for network flow problems with convex arc costs.* SIAM Journal on Control and Optimization **25**, pp. 1219–1243.

[8] Bregman, L.M. (1967), *The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming.* USSR Computational Mathematics and Mathematical Physics **7**, pp. 200–217.

[9] Bruynooghe, M., Gibert, A. and Sakarovitch, M. (1969), *Une méthode d'affectation du trafic.* In: Fourth International Symposium on the Theory of Traffic Flow, Karlsruhe, 1968, W. Lentzback and P. Baron (eds.), Beiträge zur Theorie des Verkehrsflusses Strassenbau und Strassenverkehrstechnik Heft 86, Herausgeben von Bundesminister fur Verkehr, Abteilung Strassenbau, Bonn, pp. 198–204.

[10] Bureau of Public Roads (1964), *Traffic Assignment Manual.* U.S. Government Printing Office, Washington, D.C.

[11] Censor, Y. (1981), *Row-action methods for huge and sparse systems and their applications.* SIAM Review, **23**, pp. 444–466.

[12] Curet, N.D. (1992), *On the dual coordinate ascent approach for nonlinear networks.* Computers and Operations Research **20**, pp. 133–140.

[13] Dafermos, S. (1980), *Traffic equilibrium and variational inequalities.* Transportation Science **14**, pp. 42–54.

[14] Dafermos, S.C. and Sparrow, F.T. (1969), *The traffic assignment problem for a general network.* Journal of Research of the National Bureau of Standards **73B**, pp. 91–118.

[15] Everett, (1963), *Generalized Lagrange multiplier method for solving problems of optimum allocation of resources.* Operations Research **11**, pp. 399–417.

[16] Frank, M. and Wolfe, P. (1956), *An algorithm for quadratic programming.* Naval Research Logistics Quarterly **3**, pp. 95–110.

[17] Hearn, D.W., Lawphongpanich, S. and Nguyen, S. (1984), *Convex programming formulations of the asymmetric traffic assignment problem.* Transportation Research **18B**, pp. 357–365.

[18] Hearn, D.W., Lawphongpanich, S. and Ventura, J.A. (1987), *Restricted simplicial decomposition: Computation and extensions.* Mathematical Programming Study **31**, pp. 99–118.

[19] von Hohenbalken, B. (1975), *A finite algorithm to maximize certain pseudoconcave functions on polytopes.* Mathematical Programming **9**, pp. 189–206.

[20] von Hohenbalken, B. (1977), *Simplicial decomposition in nonlinear programming algorithms.* Mathematical Programming **13**, pp. 49–68.

[21] Kennington, J.L., Helgason, R.V. (1980). Algorithms for Network Programming. John Wiley & Sons, New York, NY.

[22] Lamond, B. and Stewart, N.F. (1981), *Bregman's balancing method.* Transportation Research **15B**, pp. 239–248.

[23] Larsson, T. and Migdalas, A. (1990), *An algorithm for nonlinear programs over Cartesian product sets.* Optimization **21**, pp. 535–542.

[24] Lawphongpanich, S. and Hearn, D.W. (1984), *Simplicial decomposition of the asymmetric traffic assignment problem.* Transportation Research **18B**, pp. 123–133.

[25] LeBlanc, L.J., Morlok, E. and Pierskalla, W.P. (1975), *An efficient approach to solving the road network equilibrium traffic assignment problem.* Transportation Research **9**, pp. 308–318.

[26] Luenberger, D.G. (1984). Linear and Nonlinear Programming. Second Edition, Addison-Wesley, Reading, MA.

[27] Meyer, G.G.L. (1974), *Accelerated Frank-Wolfe algorithms.* SIAM Journal on Control **12**, pp. 655–663.

[28] Netter, M. and Sender, J.G. (1970), *Equilibre offre-demande et tarification sur un réseau de transport.* Institut de Recherche des Transport, Arcueil, France.

[29] Nguyen, S. (1974), *An algorithm for the traffic assignment problem.* Transportation Science, **8**, pp. 302–216.

[30] Nguyen, S. and Dupuis, C. (1984), *An efficient method for computing traffic equilibria in networks with asymmetric transportation costs.* Transportation Science **18**, pp. 185–202.

[31] Ortega, J.M. and Rheinboldt, W.C. (1970). Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, New York, NY.

[32] Patriksson, M. (1990), *The traffic assignment problem—theory and algorithms.* Report LiTH-MAT-R-90-29, Linköping Institute of Technology, Department of Mathematics, Linköping, Sweden.

[33] Patriksson, M. (1991), *Partial linearization methods in nonlinear programming.* Report LiTH-MAT-R-91-11, Linköping Institute of Technology, Department of Mathematics, Linköping, Sweden. Forthcoming in Journal of Optimization Theory and Applications **78**, 1993.

[34] Patriksson. M. (1991), *A unified description of iterative algorithms for traffic equilibria.* Report LiTH-MAT-R-91-35, Linköping Institute of Technology, Department of Mathematics, Linköping, Sweden. Forthcoming in European Journal of Operational Research.

[35] Petersen, E.R. (1975), *A primal-dual traffic assignment algorithm.* Management Science **22**, pp. 87–95.

[36] Pigou, A.C. (1920). The Economics of Welfare. MacMillan & Co., London.

[37] Smith, M.S. (1979), *The existence, uniqueness and stability of traffic equilibria.* Transportation Research **13B**, pp. 295–304.

[38] Wardrop, J.G. (1952), *Some theoretical aspects of road traffic research.* Proceedings of the Institution of Civil Engineering, Part II, **1**, pp. 325–378.

[39] Wolfe, P. (1970), *Convergence theory in nonlinear programming.* In: Integer and nonlinear programming, J. Abadie (ed.), North-Holland, Amsterdam, pp. 1–36.

[40] Zenios, S.A. and Mulvey, J.M. (1985/6), *Relaxation techniques for strictly convex network problems.* Annals of Operations Research **5**, pp. 517–538.

[41] Zenios, S.A. and Mulvey, J.M. (1988), *A distributed algorithm for convex network optimization problems.* Parallel Computing, **6**, pp. 45–56.

| Notation | Definition |
|---|---|
| $\mathcal{N}$ | set of nodes |
| $\mathcal{A}$ | set of links |
| $\mathcal{C}$ | set of commodities; defined by the set of origins |
| $o_k$ | origin of commodity $k$ |
| $\mathcal{D}_k$ | set of destination nodes for commodity $k$ |
| $\mathcal{W}_i$ | subset of nodes being terminal nodes of links initiated at node $i$ |
| $\mathcal{V}_i$ | subset of nodes being initial nodes of links terminating at node $i$ |
| $r_{ki}$ | desired flow for destination $i$ in commodity $k$ |
| $f_{ij}^k$ | flow on link $(i,j)$ for commodity $k$ |
| $f_{ij}$ | total flow on link $(i,j)$; equals $\sum_{k \in \mathcal{C}} f_{ij}^k$ |
| $t_{ij}(f_{ij})$ | link performance function for link $(i,j)$ |

Table 1: Notation

| Iter. | P.L. | F.-W. | C.P. | S.D. |
|---|---|---|---|---|
| 0 | 125 500.00 | 125 500.00 | 125 500.00 | 125 500.00 |
| 1 | 87 162.65 | 94 253.37 | 94 253.37 | 94 253.38 |
| 2 | 85 192.95 | 86 847.30 | 86 847.30 | 86 560.91 |
| 3 | 85 035.47 | 86 447.56 | 86 294.06 | 85 895.14 |
| 4 | 85 028.98 | 86 229.62 | 86 082.69 | 85 513.68 |
| 5 | 85 028.20 | 85 939.96 | 85 963.00 | 85 215.00 |
| 6 | 85 028.11 | | | 85 047.23 |
| 7 | | | | 85 028.07 |
| 8 | | | | 85 028.07 |
| 10 | | 85 555.79 | 85 210.60 | |
| 15 | | 85 392.72 | 85 061.23 | |
| 20 | | 85 329.47 | 85 035.67 | |
| 25 | | 85 273.72 | 85 030.39 | |
| 30 | | 85 241.00 | 85 028.53 | |
| 35 | | 85 212.33 | 85 028.14 | |
| 40 | | 85 102.52 | | |

Table 2: Comparisons between various methods for the Nguyen/Dupuis problem

| Iter. | P.L. | F.-W. |
|---:|---|---|
| 0 | 81 070.00 | 81 070.00 |
| 1 | 37 076.17 | 46 845.00 |
| 2 | 30 783.53 | 40 126.25 |
| 3 | 29 975.64 | 36 014.45 |
| 4 | 29 555.77 | 32 660.24 |
| 5 | 29 499.12 | 31 538.79 |
| 6 | 29 474.68 | 30 830.22 |
| 7 | 29 469.55 | 30 452.90 |
| 8 | 29 465.46 | 30 155.26 |
| 9 | 29 463.55 | 29 896.63 |
| 10 | 29 462.90 | 29 814.75 |
| 20 | | 29 533.08 |
| 30 | | 29 505.26 |
| 40 | | 29 498.32 |
| 50 | | 29 495.36 |

Table 3: Comparisons between partial linearization and Frank-Wolfe for the grid network