A Note on Bar-Gera’s Algorithm for the Origin-Based Traffic Assignment Problem

Yu (Marco) Nie
Department of Civil and Environmental Engineering
Northwestern University, y-nie@northwestern.edu, http://www.civil.northwestern.edu/people/nie.html

Recently Bar-Gera (2002) proposed a quasi-Newton method for the origin-based formulation of the user equilibrium traffic assignment problem. This note shows that Bar-Gera’s algorithm may generate negative second-order derivative, leading to a “wrong search direction” which compromises the overall convergence performance. We prove that this shortcoming can be overcome by approximating the second-order derivative with Bertsekas’s upper bound (Bertsekas, Gafni & Gallager 1984). The revised algorithm not only fixes a theoretical flaw, but indeed demonstrates more satisfying computational performance in numerical experiments. This note also offers a rigorous derivation of optimality conditions that synthesizes the results of Gallager (1977), Bertsekas et al. (1984), and Bar-Gera (2002).

Key words: Quasi-Newton method; user equilibrium; traffic assignment; origin-based formulation

1. Introduction
Searching for highly precise solutions for the static user equilibrium (UE) traffic assignment problem (TAP) (Wardrop 1952, Beckmann, McGuire & Winsten 1956) has received some revived interest recently. Traditionally, this problem was formulated and solved using a tree-based (e.g. LeBlanc, Morlok & Pierskalla 1975, Hearn, Lawphongranich & Ventura 1985, Fukushima 1985) or route-based (e.g. Gibert 1968, Dafermos & Sparrow 1969, Leventhal, Nemhauser & Trotter 1973, Bertsekas 1976, Bertsekas & Gafni 1983, Larsson & Patriksson 1992, Jayakrishnan, Tsai, Prashker & Rajadhyaksha 1994) approach. However, a bush-based approach, which represents a compromise between tree- and route-based approaches, may render more efficient solution procedures. This approach restricts traffic assignment only to acyclic sub-networks rooted at origins (or destinations), called bush, and iterates between two sub-problems: expand or trim the bushes and equilibrate flows on them. While the concept of bush may be traced back to Dial’s celebrated work on logit traffic assignment (Dial 1971), operationalizing it for the UE-TAP is a relatively new development put forth by Bar-Gera (2002) and Dial(2006). Bush-based algorithm may operate in the space of route or origin-based flows. The focus of this note is on Bar-Gera’s algorithm for the origin-based reformulation of the UE-TAP. The reader is referred to Nie (2010) for a more detailed review of bush-based algorithms.

The origin-based flows in Bar-Gera’s algorithm are represented by proportions of traffic arriving at each node from its predecessor links (called approach proportion). As Bar-Gera noted (1999), a similar approach had been exploited in the algorithms of Gallager (1977) and Bertsekas et al. (1984). Bar-Gera’s implementation gains its efficiency in the method developed to expand acyclic sub-networks, based on the notion of maximum contributing costs, and in the use of last common node (LCN) reduction to improve the approximation of second-order derivatives (cf. Section 3.2 for more details). However, it appears that Bar-Gera’s original algorithm contains a subtle yet non-trivial flaw which might substantially affects its convergence performance. The present note is devoted to the explanation and resolution of this issue.

The problem identified in Bar-Gera’s algorithm arises from one of its key features: using LCN reduction to calculate second-order derivatives. Essentially, Bar-Gera’s algorithm employs a lower
bound approximation to compute the second-order derivative of the Beckmann function (Beckmann et al. 1956) with respect to \textit{approach proportions}. This approximation may yield negative second-order derivatives when combined with the LCN reduction, which in turn leads to a “wrong search direction”\footnote{We call it a “wrong” direction because a descent direction scaled by the negative second-order derivative would point to the opposite direction.} in Newton-type methods. Section 4 provides an illustrative example. The potential problem may not manifest in Bar-Gera’s tests, mainly because his algorithm bounds the second order derivative from below by a positive and arbitrarily selected small number, albeit for a different reason (cf. Section 4 for details). Although this treatment would suppress negative second-order derivatives, it is likely to result in an imprecise search direction and hence a degraded convergence behavior.

The objective of this note is to overcome the problem of wrong search direction using an upper bound approximation of the second-order derivative. This result is first shown analytically, and then its improvement is demonstrated using numerical results. The adopted upper bound was originally derived in Bertsekas et al. (1984), though they did not specifically consider the LCN reduction.

In order to reveal the nature of the wrong search direction problem and to derive an upper bound of the second-order derivative, we provide an alternative derivation of the first-order optimality for the origin-based TAP formulation along the line of Gallager (1977) and Bertsekas et al. (1984). This derivation connects the work of Gallager and Bertsekas to Bar-Gera’s. The framework of the resulting algorithm remains unchanged from that of Bar-Gera (2002); the only difference is in the formula used to evaluate second-order derivatives.

The rest of the note is organized as follows. Section 2 briefly reviews the origin-based formulation. Section 3 derives first and second-order derivatives for Bar-Gera’s transformation of origin-based TAP, using Gallager’s approach. Section 4 highlights the wrong search direction problem and provides a solution. Section 5 reports experimental results and Section 6 offers conclusions.

2. The origin-based formulation
The origin-based formulation investigated in this note can be viewed as a variant of Beckmann’s seminal formulation, which, unlike the path-link formulation made popular by Dafermos (1968), relies on the link-node flow conservation law. Beckmann’s formulation may be written using origin-based link flows as follows (cf. Appendix for a list of notation):

\[
\min z(x) = \sum_{ij} \int_0^{x_{ij}} t_{ij}(w)dw \\
\text{subject to:}
\sum_{i \in O(j)} x_{rij} - \sum_{i \in L(j)} x_{rij} = q_j, \forall j \in N, r \in R \\
\sum_r x_{ij} = x_{ij}, x_{ij} \geq 0, \forall (i,j) \in A, r \in R,
\]

where
A and N are sets of links and nodes, respectively; 
R and S are sets of origin and destination nodes, respectively; 
x_{ij} is the flow on link ij contributed by all O-D pairs rs originating at r; 
x_{ij} is the total flow on link ij; 
t_{ij}(\cdot) is a strictly positive and increasing function of x_{ij};
$O(j)$ and $I(j)$ denote the set of successor(s) and predecessor(s) of $j$, respectively;

\[
q^r_j = \begin{cases} 
\sum_s d_{rs} & j = r \\
-d_{rs} & j = s \\
0 & \text{otherwise} 
\end{cases} \quad \forall j \in A, r \in R, s \in S; \quad \text{and} 
\]

$d_{rs}$ is the travel demand departing from origin $r$ for destination $s$.

Let $u^r_j$ be the multiplier associated with node $j$ and origin $r$ (Constraint (2)). The KKT conditions of this origin-based formulation are the feasible conditions (2-4) plus the following complementary condition

\[
t_{ij}(x_{ij}) + u^r_i - u^r_j \geq 0, \forall (i, j) \in A, r \in R \\
x^r_{ij}[t_{ij}(x_{ij}) + u^r_i - u^r_j] = 0, \forall (i, j) \in A, r \in R
\]

Problem (11)-(13) can be decomposed with respect to origins, in the spirit of the Gauss-Seidel method. Namely, we can separately consider the following problem for each origin $r$:

\[
\min z_r(x) = \sum_{ij} \int_0^{x^r_{ij} + x'^r_{ij}} t_{ij}(w) dw \\
\text{subject to :} \\
\sum_{i \in I(j)} x^r_{ij} - \sum_{l \in O(j)} x^r_{jl} = q^r_j, \forall j \in N \\
\sum_{o \in R, o \neq r} x^o_{ij} = x^r_{ij}, x'^r_{ij} \geq 0, \forall (i, j) \in A
\]

where $x'^r_{ij}$ are flows contributed by all O-D pairs $os, \forall o \neq r$. When each decomposed subproblem is solved, $x'^r_{ij}$ is treated as constant background traffic. Keeping this in mind, we suppress the origin index and concentrate on a single origin case for notational convenience. The above problem is rewritten as:

\[
\min z(x) = \sum_{ij} \int_0^{x_{ij}} t_{ij}(w) dw \\
\text{subject to :} \\
\sum_{i \in I(j)} x_{ij} - \sum_{l \in O(j)} x_{jl} = q_j, \forall j \in N \\
x_{ij} \geq 0, \forall (i, j) \in A
\]

where $q_j$ is traffic input from the origin terminating at $j$. The complementary condition for the above problem is

\[
t_{ij}(x_{ij}) + u_i - u_j \geq 0, \forall (i, j) \in A \\
x_{ij}[t_{ij}(x_{ij}) + u_i - u_j] = 0, \forall (i, j) \in A
\]

where $u_i$ is the minimum travel time from the origin to $i$.

**Proposition 1 (Acyclicity of user equilibrium).** For the single-origin formulation (10) - (12), links that have positive flow at user equilibrium never form a cycle.

Proof: See Lemma 1 - 3 in (Bar-Gera 2002) or Lemma 3 (Dial 2006). Also see pp. 154 - 155, Newell (1980). $\square$

Thus, the UE link flows of a single-origin TAP form an acyclic sub-network, which is called a bush in Dial (1971). Nie (2010) defines a bush formally as follows:
Definition 1 (Bush). A directed network is called a bush rooted at \( r \) if it (1) is acyclic, (2) has at least one route from \( r \) to every other node. Thanks to acyclicity, nodes in a bush can be ordered according to their topological distances, which makes efficient recursive calculation possible (cf. Section 3.2). The topological distance of a node in a bush is defined in the following.

Definition 2 (Topological Distance). Let the length of every link in a bush be 1. The topological distance of node \( j \), denoted as \( \pi_j \), is the maximum distance from the origin to \( j \).

Definition 3 (Ascending (Descending) Pass). An ascending (descending) pass is a sequential visit to each node of a bush following the increasing (decreasing) order of topological distance.

While properly constructing bushes is critical to the success of origin-based algorithms, it is not the focus of this note. The reader is referred to Nie (2010) for discussions of that subject. Hereafter we shall assume acyclicity holds unless otherwise specified.

3. First- and second-order derivatives

Assuming acyclicity, the single-origin formulation (10-12) can be transformed using routing variables as first shown in Gallager (1977). We note that these routing variables are equivalent to approach proportions in Bar-Gera (2002).

Let \( \eta_j \) be the total flow entering node \( j \) and \( \phi_{ij} \) denote the proportion of \( \eta_j \) that uses link \( ij \). The formulation (10) - (12) is rewritten as

\[
\min z(\phi) = \sum_{ij} \int x_{ij} t_{ij}(w) dw \tag{15}
\]

subject to:

\[
x_{ij} = \eta_j \phi_{ij}, \forall j \tag{16}
\]

\[
\eta_j = q_j + \sum_{l \in O(j)} \eta_l \phi_{jl}, \forall j \tag{17}
\]

\[
\sum_{i \in I(j)} \phi_{ij} = 1, \forall j; \phi_{ij} \geq 0, \forall ij \tag{18}
\]

3.1. Establish the equivalent UE condition

In Problem (15) - (18), \( \phi_{ij} \) is used as the main free variable, and \( q_j \) is treated as an auxiliary free variable for deriving a recursive formula. Also, \( \eta_j \) is a function of \( q_j \) and \( \phi_{ij} \); \( x_{ij} \) is a function of \( \eta_j \) and \( \phi_{ij} \); and the objective function \( z \) is a direct function of link flows \( x_{ij} \). In order to establish the KKT condition, we need to obtain the gradient of \( z(\phi) \) with respect to the routing variables \( \phi \). To this end, we first examine some useful properties of Constraint (17), and then derive the derivative of the arriving flows \( \eta_j \) with respect to free variables \( q \) and \( \phi \).

Let us rewrite Equation (17) in a matrix form as

\[
\eta = q + \Phi \eta \tag{19}
\]

where \( \eta \) and \( q \) are \( n \times 1 \) vectors (\( n = |N| \) is the number of nodes) and \( \Phi \) is a \( n \times n \) matrix. Equation (19) can be reorganized as

\[
\eta = \Lambda q \tag{20}
\]

where \( \Lambda = (I - \Phi)^{-1} \) is also an \( n \times n \) matrix. Note that \( (I - \Phi) \) has an inverse because (19) always has a unique solution for any \( q > 0 \) (note that (19) can always be solved by a descending topological pass for a given \( \phi \)). Taking the derivative of \( \eta_j \) with respect to \( q_l \) in (20), we get

\[
\frac{\partial \eta_j}{\partial q_l} = \lambda_{jl} \tag{21}
\]

where \( \lambda_{jl} \) is an entry in \( \Lambda \).
Lemma 1. The derivative of $\eta_j$ with respect to $q_k$ can be written as

$$\frac{\partial \eta_j}{\partial q_k} = \begin{cases} 1 & j = k \\ \sum_{l \in O(j)} \frac{\partial \eta_l}{\partial q_k} & \pi_j < \pi_k \\ 0 & \pi_j > \pi_k \end{cases}$$

or

$$\frac{\partial \eta_j}{\partial q_k} = \begin{cases} 1 & j = k \\ \sum_{l \in I(k)} \frac{\partial \eta_l}{\partial q_k} & \pi_j < \pi_k \\ 0 & \pi_j > \pi_k \end{cases}$$

Proof: Taking the derivative with respect to $q_k$ on both sides of Equation (17) yields

$$\frac{\partial \eta_j}{\partial q_k} = \frac{\partial q_j}{\partial q_k} + \sum_{l \in O(j)} \frac{\partial \eta_l}{\partial q_k}$$

When $j = k$, the second term in the above equation diminishes, because changing the terminating flow $q_k$ can only affect arriving flows of upstream nodes. The derivative is thus 1 in this case. When $\pi_j < \pi_k$, the first term becomes zero, and thus

$$\frac{\partial \eta_j}{\partial q_k} = \sum_{l \in O(j)} \frac{\partial \eta_l}{\partial q_k}$$

When $\pi_j > \pi_k$, changes of $q_k$ will affect neither $\eta_j$ nor $q_j$. Thus $\frac{\partial \eta_j}{\partial q_k} = 0$, which proves that the recursive relationship (22) holds.

To prove the second part, we note the following from Equations (19) and (20):

$$\Lambda(I - \Phi) = I$$

Multiplying the $j$th row in $\Lambda$ with the $k$th column in $I - \Phi$, we obtain

$$\lambda_{jk} - \sum_{l \in N, l \neq k} \lambda_{jl} \phi_{lk} = \begin{cases} 1 & j = k \\ 0 & j \neq k \end{cases}$$

Using (21), the above equation can be written as

$$\frac{\partial \eta_j}{\partial q_k} = \begin{cases} 1 + \sum_{l \in N, l \neq k} \frac{\partial \eta_l}{\partial q_k} & j = k \\ \sum_{l \in N, l \neq k} \frac{\partial \eta_l}{\partial q_k} & j \neq k \end{cases}$$

When $j = k$, $\sum_{l \in N, l \neq k} \frac{\partial \eta_l}{\partial q_k} = 0$ because $\pi_l < \pi_j$. When $\pi_j > \pi_k$, $\frac{\partial \eta_j}{\partial q_k}$ is zero due to acyclicity. Finally if $\pi_j < \pi_k$, Equation (28) is simplified as

$$\frac{\partial \eta_j}{\partial q_k} = \sum_{l \in I(k)} \frac{\partial \eta_l}{\partial q_k}$$

This proves the second recursive relationship.

Remark: Equations (22) and (23) are equivalent to Equations (2.17) and (2.18) in Bar-Gera (1999), although they are obtained using different approaches. Also, Bar-Gera’s $\chi_{i \rightarrow j}$ equals $\lambda_{ij}$ in our notation, which was nicely interpreted in Bar-Gera (1999) as the proportion of traffic arriving at node $j$ through $i$. 

□
Lemma 2. The derivative of $\eta_j$ with respect to routing variable $\phi_{km}$ can be written as

$$\frac{\partial \eta_j}{\partial \phi_{km}} = \frac{\partial \eta_j}{\partial q_k} \eta_m$$  \hspace{1cm} (30)

Proof: Taking derivatives on both sides of Equation (17) with respect to $\phi_{km}$, we obtain

$$\frac{\partial \eta_j}{\partial \phi_{km}} = \sum_{l \in O(j)} \left( \frac{\partial \eta_l}{\partial \phi_{km}} \phi_{jl} + \eta_l \frac{\partial \phi_{jl}}{\partial \phi_{km}} \right) = \sum_{l \in O(j)} \frac{\partial \eta_l}{\partial \phi_{km}} \phi_{jl} + \eta_m \delta_{jk}$$  \hspace{1cm} (31)

where $\delta_{jk} = 1$ if $j = k$ and 0 otherwise. For any fixed $k, m$, let $\eta' = [\cdots, \frac{\partial \eta_j}{\partial \phi_{km}}, \cdots]^T$ and $\hat{\eta} = [\cdots, \eta_m \delta_{jk}, \cdots]^T$ ($\eta'$ and $\hat{\eta}$ are $n \times 1$ vectors). Similar to Equation (19), we can rewrite (31) in a matrix form

$$\eta' = \hat{\eta} + \Phi \eta'$$  \hspace{1cm} (32)

Thus, $\eta'$ may be written as

$$\eta' = \Lambda \hat{\eta}$$  \hspace{1cm} (33)

Thus we have

$$\frac{\partial \eta_j}{\partial \phi_{km}} = \eta'_j = \sum_{i \in N} \frac{\partial \eta_j}{\partial q_i} \hat{\phi}_{ij} = \sum_{i \in N} \frac{\partial \eta_j}{\partial q_i} \eta_m \delta_{ik} = \frac{\partial \eta_j}{\partial q_k} \eta_m$$  \hspace{1cm} (34)

The last equality holds because $\delta_{ik} = 0$ for all $l \neq k$. This completes the proof. □

We are now ready to present the main result, the gradient of $z(\phi)$ with respect to routing variables $\phi_{ij}$.

Proposition 2. In the optimization problem (15) - (18), the gradient of $z(\phi)$ with respect to $\phi$ can be calculated using the following recursive formula:

$$\frac{\partial z(\phi)}{\partial \phi_{km}} = \eta_m \left( t_{km} + \frac{\partial z(\phi)}{\partial q_k} \right)$$  \hspace{1cm} (35)

$$\frac{\partial z(\phi)}{\partial q_k} = \sum_i \phi_{ik} \left( t_{ik} + \frac{\partial z(\phi)}{\partial q_i} \right)$$  \hspace{1cm} (36)

$$\frac{\partial z(\phi)}{\partial q_r} = 0$$  \hspace{1cm} (37)

Proof: From the chain rule, we have

$$\frac{\partial z(\phi)}{\partial \phi_{km}} = \sum_{ij} t_{ij} \left( \frac{\partial x_{ij}}{\partial \phi_{km}} + \phi_{ij} \frac{\partial x_{ij}}{\partial \phi_{km}} \right) = \eta_m t_{km} + \sum_{ij} t_{ij} \phi_{ij} \frac{\partial \eta_j}{\partial \phi_{km}}$$  \hspace{1cm} (38)

Using Lemma 2, Equation (38) can be simplified as

$$\frac{\partial z(\phi)}{\partial \phi_{km}} = \eta_m t_{km} + \sum_{ij} t_{ij} \phi_{ij} \frac{\partial \eta_j}{\partial q_k} \eta_m = \eta_m \left( t_{km} + \sum_{ij} t_{ij} \phi_{ij} \frac{\partial \eta_j}{\partial q_k} \right) = \eta_m \left( t_{km} + \frac{\partial z(\phi)}{\partial q_k} \right)$$  \hspace{1cm} (39)

This leads to (35). We proceed to show how $\frac{\partial z(\phi)}{\partial q_k}$ can be evaluated recursively. Note that

$$\frac{\partial z(\phi)}{\partial q_k} = \sum_{ij} t_{ij} \frac{\partial x_{ij}}{\partial q_k} = \sum_{ij} t_{ij} \left( \frac{\partial x_{ij}}{\partial \phi_{ij}} \frac{\partial \phi_{ij}}{\partial q_k} + \frac{\partial x_{ij}}{\partial \eta_j} \frac{\partial \eta_j}{\partial q_k} \right) = \sum_{ij} t_{ij} \frac{\partial x_{ij}}{\partial \eta_j} \frac{\partial \eta_j}{\partial q_k} = \sum_{ij} t_{ij} \phi_{ij} \frac{\partial \eta_j}{\partial q_k}$$  \hspace{1cm} (40)
Network flow is not UE because route 1-2-3-4-5 is shorter than 1-5.

Figure 1 An illustration of the equivalent UE condition

According to Lemma 1 (Equation (23)), any link $ij$ may be grouped into each of the three classes: links ending at $k$ ($j = k$), links upstream of $k$ ($\pi_j < \pi_k$), and links downstream of $k$ ($\pi_j > \pi_k$) or with the same topological order but different ending nodes ($\pi_j = \pi_k, j \neq k$). Accordingly,

$$
\frac{\partial z(\phi)}{\partial q_k} = \sum_{ij, jk} t_{ij} \phi_{ij} + \sum_{ij \in A, \pi_j < \pi_k} \phi_{jk} \sum_{l \in I(k)} \frac{\partial \eta_l}{\partial q_k} \phi_{lk} + \sum_{ij \in A, \pi_j > \pi_k} \phi_{ij} \sum_{l \in I(k)} t_{ij} \phi_{ij} \times 0 \quad (41)
$$

Rearranging the summation in the above equation we obtain

$$
\frac{\partial z(\phi)}{\partial q_k} = \sum_{l \in I(k)} \phi_{lk} \left( t_{lk} + \sum_{ij \pi_j < \pi_k} t_{ij} \phi_{ij} \frac{\partial \eta_l}{\partial \eta_j} \right) \quad (42)
$$

It follows from Equation (42) that the second term in the above Equation is simply $\partial z(\phi)/\partial \eta_j$. This leads to Equation (36). Finally, it is obvious that $\partial z(\phi)/\partial q_r = 0$ ($r$ is the origin), again because $q_r$ only affects upstream nodes.

We proceed to examine the first-order optimality conditions. The complementary condition associated with the non-negative constraint implies

$$
\phi_{km} > 0 \quad \Rightarrow \quad \frac{\partial z(\phi)}{\partial \phi_{km}} = \mu_m; \quad \frac{\partial z(\phi)}{\partial \phi_{km}} \geq \mu_m \rightarrow \phi_{km} = 0; \forall m \neq r \quad (43)
$$

where $\mu_m$ is the Lagrangian multiplier associated with (18). As noted in Gallager (1977), however, Condition (43) is not sufficient to ensure the UE condition essentially because $z(\phi)$ is not convex in $\phi$. Evidently, Condition (43) will be automatically satisfied whenever $\eta_m = 0$. Figure 1 gives an example in which Condition (43) is satisfied at all nodes while the network flow pattern is not UE.

Let us define

$$
\tilde{v}_k = \frac{\partial z(\phi)}{\partial q_k}; \quad v_{km} = t_{km} + \tilde{v}_k \quad (44)
$$

2 This important recursive relationship was derived using the following intuitive reasoning in Gallager (1977). Note that any tiny change $\epsilon$ in $q_i$ will change the flow on each predecessor link $ik$ by $\epsilon \phi_{ik}$. This corresponds to the change of $\epsilon \phi_{ik} t_{ik}$ in the objective function value. If $i$ is not the origin, then $\epsilon \phi_{ik}$, which now is interpreted as the change in $q_i$, will change the objective function by $\epsilon \phi_{ik} \partial z(\phi)/\partial q_i$ from node $i$ backward. Our proof formalizes this idea.
It follows that
\[ \frac{\partial z(\phi)}{\partial \phi_{km}} = v_{km} \eta_m \] 

(45)

\( v_{km} \) and \( \bar{v}_k \) can be recursively evaluated using
\[ v_{km} = t_{km} + \bar{v}_k; \bar{v}_k = \sum_{i \in I(k)} \phi_{ik} v_{ik} ; \bar{v}_r = 0 \] 

(46)

**Theorem 1 (A sufficient UE condition).** A feasible solution of the optimization problem \([13] - [18]\) is user-equilibrium if the following condition is satisfied.
\[ v_{km} - u_m \geq 0; \phi_{km}(v_{km} - u_m) = 0, \forall k, m \neq r \] 

(47)

where \( u_m = \min \{v_{km}, \forall k \in I(m)\} \)

Proof: cf. Theorem 3, (Gallager 1977), also Lemma 5, (Bar-Gera 2002).

□

**Remark:** At UE, \( u_m \) is the minimum travel time from the origin to node \( m \) (cf. Condition \([13] - [14]\)), and \( v_{km} \) is interpreted as the average travel time from the origin to node \( m \) through link \( km \). Therefore, a link \( km \) is used only when the average travel time arriving at node \( m \) through \( km \) is the minimum among all \( m \) incoming links. Note that Condition (47) is not satisfied in the example shown in Figure 1. Hereafter Condition (47) is used to ensure the UE condition.

### 3.2. Approximate second order derivatives

We proceed to derive the second-order derivative of \( z(\phi) \). Only the diagonal elements of the Hessian matrix are considered because they are sufficient for quasi-Newton-type methods.

\[ \frac{\partial^2 z(\phi)}{\partial \phi_{km}^2} = \frac{\partial}{\partial \phi_{km}} \left[ \eta_m \left( t_{km} + \frac{\partial z(\phi)}{\partial q_k} \right) \right] = \frac{\partial \eta_m}{\partial \phi_{km}} \left( t_{km} + \frac{\partial z(\phi)}{\partial q_k} \right) + \eta_m \left( \frac{\partial t_{km}}{\partial \phi_{km}} + \frac{\partial^2 z(\phi)}{\partial q_k \partial \phi_{km}} \right) \] 

(48)

Since \( km \) is a link upstream of \( m \), the change of \( \phi_{km} \) will not change the total arriving flow at \( m \) due to acyclicity, i.e., \( \frac{\partial \eta_m}{\partial \phi_{km}} = 0 \). Also note
\[ \frac{\partial t_{km}}{\partial \phi_{km}} = t'_{km} \frac{\partial x_{km}}{\partial \phi_{km}} = \eta_m t'_{km}, \] 

(49)

where \( t'_{ij} = \frac{dt_{ij}(x_{ij})}{dx_{ij}} \) and
\[ \frac{\partial^2 z(\phi)}{\partial q_k \partial \phi_{km}} = \frac{\partial}{\partial q_k} \frac{\partial z(\phi)}{\partial \phi_{km}} = \frac{\partial}{\partial q_k} \left[ \eta_m \left( t_{km} + \frac{\partial z(\phi)}{\partial q_k} \right) \right] = \eta_m \frac{\partial^2 z(\phi)}{\partial q_k^2}. \] 

(50)

The last equality holds because acyclicity leads to \( \frac{\partial \eta_m}{\partial q_k} = 0, \frac{\partial t_{km}}{\partial q_k} = 0 \). Thus
\[ \frac{\partial^2 z(\phi)}{\partial^2 \phi_{km}} = \eta_m^2 \left( t'_{km} + \frac{\partial^2 z(\phi)}{\partial q_k^2} \right) \] 

(51)

Recalling Equation (36)
\[ \frac{\partial^2 z(\phi)}{\partial^2 q_k} = \frac{\partial}{\partial q_k} \left[ \sum_{i \in I(k)} \phi_{ik} \left( t_{ik} + \frac{\partial z(\phi)}{\partial q_k} \right) \right] = \sum_{i \in I(k)} \phi_{ik} \left( t'_{ik} \phi_{ik} + \frac{\partial^2 z(\phi)}{\partial q_i \partial q_k} \right) = \sum_{i \in I(k)} \phi_{ik}^2 t'_{ik} + \sum_{i \in I(k)} \phi_{ik} \frac{\partial^2 z(\phi)}{\partial q_i \partial q_k} \] 

(52)
we have (note $\frac{\partial t_{ik}}{\partial q_i} = 0$ due to acyclicity)
\[
\frac{\partial^2 z(\phi)}{\partial q_i \partial q_k} = \frac{\partial}{\partial q_i} \left[ \sum_{j \in I(k)} \phi_{jk} \left( t_{jk} + \frac{\partial z(\phi)}{\partial q_j} \right) \right] = \sum_{j \in I(k)} \phi_{jk} \frac{\partial^2 z(\phi)}{\partial q_i \partial q_j}
\] (53)

Consequently
\[
\frac{\partial^2 z(\phi)}{\partial^2 q_k} = \sum_{i \in I(k)} \phi_{ik} t_{ik} + \sum_{i \in I(k)} \sum_{j \in I(k)} \phi_{ik} \phi_{jk} \frac{\partial^2 z(\phi)}{\partial q_i \partial q_j} = \sum_{i \in I(k)} \phi_{ik} \frac{\partial^2 z(\phi)}{\partial^2 q_i} + \sum_{j \in I(k), j \neq i} \phi_{ik} \phi_{jk} \frac{\partial^2 z(\phi)}{\partial q_i \partial q_j}
\] (54)

To develop bounds for $\frac{\partial^2 z(\phi)}{\partial^2 q_i}$, we need the following lemma.

**Lemma 3.** The second order derivative of $z(\phi)$ with respect to the auxiliary free variable $q$ can be evaluated by
\[
\frac{\partial^2 z(\phi)}{\partial q_i \partial q_j} = \sum_{km \in A} p_{km}^i p_{km}' t_{km}' \frac{\partial^2 z(\phi)}{\partial^2 q_i} = \sum_{km \in A} (p_{km}')^2 t_{km}'
\] (55)

where
\[
p_{km}^i = \phi_{km} \frac{\partial \eta_m}{\partial q_i}
\] (56)

**Proof:** The following result directly follows from Equation (40)
\[
\frac{\partial z(\phi)}{\partial q_i} = \sum_{km \in A} t_{km} \phi_{km} \frac{\partial \eta_m}{\partial q_i}
\] (57)

Thus
\[
\frac{\partial^2 z(\phi)}{\partial q_i \partial q_j} = \sum_{km \in A} \phi_{km} \left( \frac{\partial t_{km}}{\partial q_j} \frac{\partial \eta_m}{\partial q_i} + t_{km} \frac{\partial^2 \eta_m}{\partial q_i \partial q_j} \right)
\] (58)

From Equation (20) we know that $\eta$ can be expressed as a linear function of $q$. This implies that the second order derivative of $\eta$ with respect to $q$ is zero. Thus the second term in the parenthesis of (58) vanishes. Consequently,
\[
\frac{\partial^2 z(\phi)}{\partial q_i \partial q_j} = \sum_{km \in A} \phi_{km} \frac{\partial t_{km}}{\partial q_j} \frac{\partial \eta_m}{\partial q_i} = \sum_{km \in A} \phi_{km} t_{km}' \frac{\partial x_{km}}{\partial q_j} \frac{\partial \eta_m}{\partial q_i} = \sum_{km \in A} \phi_{km} t_{km}' \phi_{km} \frac{\partial \eta_m}{\partial q_j} \frac{\partial \eta_m}{\partial q_i}
\] (59)

Refer to Equation (40) for the last equality. This completes the proof. $\square$

**Remark:** As mentioned before, $\lambda_{mi} = \frac{\partial \eta_m}{\partial q_i}$ may be interpreted as the proportion of traffic arriving at $i$ through $m$. Thus $p_{km}^i = \lambda_{mi} \phi_{km}$ is the proportion of traffic arriving at $i$ through link $km$. This result is given in Bertsekas et al. (1984) without a proof.

Lemma 3 implies that $\frac{\partial^2 z(\phi)}{\partial^2 q_i} \geq 0$. Recalling Equation (54), we have the following inequality
\[
\frac{\partial^2 z(\phi)}{\partial^2 q_i} \geq \sum_{i \in I(k)} \phi_{ik} \left( t_{ik}' + \frac{\partial z(\phi)}{\partial q_i} \right)
\] (60)

This lower bound, adopted by Bar-Gera (2002) as an approximation for the second order derivative, may give negative values as explained in Section 4.
Proposition 3 (An upper bound of the second order derivative). An upper bound of the second order derivative of \( z(\phi) \) with respect to the auxiliary free variable \( q \) is given by

\[
\frac{\partial^2 z(\phi)}{\partial^2 q_k} \leq \sum_{i \in I(k)} \phi_{ik}^2 t_{ik}^t + \left( \sum_{i \in I(k)} \phi_{ik} \sqrt{\frac{\partial^2 z(\phi)}{\partial^2 q_i}} \right)^2
\] (61)

Proof: Recall the Cauchy-Schwartz inequality

\[
\sum_i x_i y_i \leq \sqrt{\sum_i x_i^2 \sum_i y_i^2} \quad (62)
\]

Replacing \( x_i \) by \( p_{km}^t \sqrt{t_{km}^t} \) and \( y_i \) by \( p_{km}^t \sqrt{t_{km}^t} \) (cf. Equation (56)) yields

\[
\sum_{km \in A} p_{km}^t p_{km}^t t_{km}^t \leq \sqrt{\sum_{km \in A} (p_{km}^t)^2 t_{km}^t \sum_{km \in A} (p_{km}^t)^2 t_{km}^t} \quad (63)
\]

Using Equation (55), this becomes

\[
\frac{\partial^2 z(\phi)}{\partial q_i \partial q_j} \leq \sqrt{\frac{\partial^2 z(\phi)}{\partial^2 q_i} \frac{\partial^2 z(\phi)}{\partial^2 q_j}} \quad (64)
\]

Applying this inequality in (54) yields

\[
\frac{\partial^2 z(\phi)}{\partial^2 q_k} \leq \sum_{i \in I(k)} \left[ \phi_{ik}^2 t_{ik}^t + \frac{\partial^2 z(\phi)}{\partial^2 q_i} \right] + \sum_{j \neq i} \phi_{ik} \phi_{jk} \sqrt{\bar{s}_j \bar{s}_i} \quad (65)
\]

Rearranging the above inequality yields the expression in (61).

Remark: This upper bound was first established by Bertsekas et al. (1984). The above derivation follows their idea but provides more details.

4. Wrong search direction and a resolution

Before looking into the wrong search direction problem, let us first explain how Bar-Gera’s quasi-Newton method (2002) is designed. To simplify the notation, define

\[
s_k = \frac{\partial^2 z(\phi)}{\partial^2 q_k}, s_{km} = t_{km}^t + s_k \quad (66)
\]

From Equation (51), the above definition implies

\[
\frac{\partial z^2(\phi)}{\partial^2 \phi_{km}} = s_{km} t_{km}^2 \quad (67)
\]

Using the lower bound defined in (60), \( s_{km} \) can be calculated using the recursive relationship

\[
s_{km} = t_{km}^t + \bar{s}_k; \bar{s}_k = \sum_i \phi_{ik}^2 s_{ik}; \bar{s}_r = 0; \quad (68)
\]

If the upper bound (65) is applied, the recursive formula reads:

\[
s_{km} = t_{km}^t + \bar{s}_k; \bar{s}_k = \sum_i \phi_{ik}^2 s_{ik} + \sum_{j \neq i} \phi_{ik} \phi_{jk} \sqrt{\bar{s}_j \bar{s}_i}; \bar{s}_r = 0; \quad (69)
\]
The key to the efficiency of Newton-type methods is simplifying the constraint structure so that projection operations can be quickly performed when needed. Similar to the gradient-projection algorithm for the route-based TAP formulation (Bertsekas & Gafni 1983, Jayakrishnan et al. 1994), Bar-Gera (2002) defines \( \bar{c} \) as a reference routing variable, i.e.,

\[
\phi_{km} = 1 - \sum_{k \in I(m), k \neq \bar{k}} \phi_{km}
\]  

(70)

Embedding this condition into the objective function, a constrained version of Equation (35) reads

\[
\frac{\partial z_c(\phi)}{\partial \phi_{km}} = \eta_m \left( t_{km} - t_{km}^* + \frac{\partial z(\phi)}{\partial q_k} - \frac{\partial z(\phi)}{\partial q_{\bar{k}}} \right) = \eta_m (c_{km} - c_{km}^*)
\]  

(71)

where \( z_c \) is the objective function defined on the space of non-reference routing variables. Bar-Gera (2002) approximated the second-order derivative of the constrained problem by

\[
\frac{\partial z_c^2(\phi)}{\partial^2 \phi_{km}} \approx \eta_m^2 (s_{km} + s_{\bar{k}m} - 2\bar{s}_{\chi(m)})
\]  

(72)

where \( \chi(m) \) represents an upstream node of \( m \) that satisfies \( \sum_i p_{i\chi(m)}^m = 1 \). \( \chi(m) \) is called the last common node of node \( m \) in Bar-Gera (2002). A sketch of Bar-Gera’s algorithm now follows:

**Step 1:** following an ascending pass (cf. Definition 2), evaluate \( \bar{c}_k \) and \( c_{km} \) using (16), and \( \bar{s}_k \) and \( s_{km} \) using (68) (lower bound approximation).

**Step 2:** following a descending pass (cf. Definition 3), update the routing variable on each incoming link to node \( m \) using

\[
\phi_{km} = \max \left( 0, \phi_{km} + \alpha \frac{\partial z_c(\phi)}{\partial \phi_{km}} / \frac{\partial z_c^2(\phi)}{\partial^2 \phi_{km}} \right), \forall k \neq \bar{k}; \phi_{km} = 1 - \sum \phi_{km}
\]  

(73)

This amounts to updating the origin-based flow on each incoming link by

\[
x_{km} = \max \left( 0, x_{km} + \alpha \eta_m \frac{\partial z_c(\phi)}{\partial \phi_{km}} / \frac{\partial z_c^2(\phi)}{\partial^2 \phi_{km}} \right), \forall k \neq \bar{k}; x_{km} = \eta_m - \sum x_{km}
\]  

(74)

where \( \frac{\partial z_c(\phi)}{\partial \phi_{km}} \) and \( \frac{\partial z_c^2(\phi)}{\partial^2 \phi_{km}} \) are computed from Equations (71) and (72), respectively. Note that \( \alpha \) in Equations (73) and (74) is a step size that can be either pre-determined or obtained from a line search.

The problem associated with using the recursive relation (68) is that \( \frac{\partial z_c(\phi)}{\partial \phi_{km}} \) defined in Equation (72) may be negative in some circumstances. Figure 2 gives such an example, in which node 10 is the last common node of node 15. The figure also demonstrates that using the recursive formula (69) can eliminate negative values.

Bar-Gera (2002) proposed to override any negative value with a tiny positive number \( \epsilon \) to guarantee convergence (cf. Equation (35), Bar-Gera 2002). In our notation, it is written as

\[
\frac{\partial z_c^2(\phi)}{\partial^2 \phi_{km}} \approx \max(\epsilon, \eta_m^2 (s_{km} + s_{\bar{k}m} - 2\bar{s}_{\chi(m)}))
\]  

(75)

This treatment may introduce unnecessary oscillations, since a tiny second-order derivative will lead to a large (and incorrect) flow shift. However, if the recursive relationship (69) is employed to evaluate \( s_{km} \) (in **Step 1**), negative \( \frac{\partial z_c^2(\phi)}{\partial^2 \phi_{km}} \) can be completely avoided. Consequently, no artificial \( \epsilon \) is involved in the algorithm. We present the result formally as follows.

\( ^3 \) \( \epsilon \) was introduced in Bar-Gera (2002) mainly for dealing with zero-flow links, for which the computed second-order derivatives are zero if BPR-type functions are used. Apparently, Bar-Gera did not anticipate that the lower bound equation (72) in conjunction with the recursive formula (68) may cause negative second-order derivatives.
Yu Nie: A Note on Bar-Gera’s Algorithm

Article submitted to Transportation Science; manuscript no. 001

Recursive calculation using Bar-Gera’s formula (lower bound)

$$(a, b) - a$$ is the routing variable \(\phi\) of a link, \(b\) is the derivative of link travel time \(t_{ik}\), \([c] - c\) is the second order derivative corresponding to either a link or a node (i.e., \(\bar{s}_{ik}, s_{ik}\)).

Legend:

Recursive calculation using the proposed formula (upper bound)

$$(a, b) + a$$ is the routing variable \(\phi\) of a link, \(b\) is the derivative of link travel time \(\bar{t}_{ik}\), \([c] + c\) is the second order derivative corresponding to either a link or a node (i.e., \(\bar{s}_{ik}, \bar{s}_{ik}\)).

Figure 2 Recursive calculation of second-order derivative using lower and upper bounds

PROPOSITION 4 (Nonnegative second-order derivative). The constrained second-order derivative \(\frac{\partial^2 z^*(\phi)}{\partial s_{km}^2}\) defined in Equation (72) is always nonnegative if: 1) link performance function \(t_{ij}(\cdot)\) is a strictly increasing function; and 2) the recursive relationship (69) is used to evaluate unconstrained second-order derivative \(s_{ikm}\).

Proof: Let \(\chi(m)\) be the last common node of \(m\). We only need to show that for any node \(i\) downstream of \(\chi(m)\)

\[
\bar{s}_i \geq \bar{s}_{\chi(m)}
\]

because for any outgoing link \(ik\), \(s_{ik} = \bar{s}_i + t'_{ik} \geq \bar{s}_i\) as per the first condition above (also cf. Equation (69)). Note that for any \(i\) with a single incoming link \(ji\), it is obvious that \(\bar{s}_i \geq \bar{s}_j\). If a node \(i\) has multiple incoming links, we claim that

\[
\bar{s}_i \geq \bar{s}_{jm}
\]

where \(j_m = \text{argmin}\{j|\bar{s}_j\}\). To see this, note that from Equation (69), we have

\[
\bar{s}_i \geq (\sum_j \phi_{ji} \sqrt{\bar{s}_j})^2 \geq (\sum_j \phi_{ji} \sqrt{\bar{s}_{jm}})^2 = \bar{s}_{jm}
\]

Therefore, if \(i\) is a node between \(m\) and \(\chi(m)\), one can always construct a path from \(i\) to \(\chi(m)\) such that \(\bar{s}_i \geq \bar{s}_{\chi(m)}\). This completes the proof.

5. Numerical results

Bar-Gera’s algorithm (2002), as well as the proposed improvement based on upper-bound approximation, are coded on top of Toolkit of Network Modeling (TNM) (Nie 2006). TNM is a suite of
Yu Nie: A Note on Bar-Gera’s Algorithm
Article submitted to Transportation Science; manuscript no. 001

reusable C++ classes for general network applications, and is adopted to simplify code development and to provide a uniform platform for comparison. Although such a general-purpose class library inevitably imposes an extra computational burden, our focus here is to reveal the difference of the methods rather than to develop the most efficient code. For convenience Bar-Gera’s method and the “new” method are hereafter referred to as BOB and NOB, respectively. All codes are run on a Windows-XP(64) workstation with two 3.00 GHz Xeon CPUs and 4G RAM. The relative gap is employed as the global convergence measure, which is defined by

\[
g_r = 1 - \frac{\sum_{rs} u_{rs} q_{rs}}{\sum_{ij} x_{ij} t_{ij}}
\]  

(76)

where \(u_{rs}\) is the minimum travel time between O-D pair \(rs\) based on the link travel time \(t_{ij}\), \(\forall ij\). Unless otherwise stated, the convergence criterion is 1E-12, and the maximum number of iterations is 200.

5.1. Preliminary test
This section compares BOB and NOB on two benchmark networks, the well-known Sioux Falls network and the 80-arc grid network employed in Dial’s recent study (2006). Figure 3 clearly shows that the origin-based algorithms achieved highly precise solutions while overcoming the tailing effect of the Frank-Wolfe algorithm (FW) widely used in practice (Frank & Wolfe 1956, LeBlanc et al. 1975). Table 1 reports the performance of the three algorithms at termination in greater details. Overall NOB and BOB behaved similarly in this experiment, although for the Dial’s network BOB

![Figure 3](image)

**Figure 3** Convergence performance of Frank-Wolfe, BOB and NOB for two small networks

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Performance of the algorithms at termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td></td>
</tr>
<tr>
<td>Networks</td>
<td>SF</td>
</tr>
<tr>
<td>Main Iterations</td>
<td>1000</td>
</tr>
<tr>
<td>CPU Time (sec)</td>
<td>0.453</td>
</tr>
<tr>
<td>Objective Value</td>
<td>42.3214</td>
</tr>
<tr>
<td>Relative gap</td>
<td>1.61E-04</td>
</tr>
</tbody>
</table>

seems marginally faster than NOB in achieving relative gaps lower than 1E-10.
5.2. Anaheim network
This section tests BOB and NOB using the Anaheim network, a medium-size network with 416 nodes, 914 links and 1406 OD pairs (Jayakrishnan et al. 1994). To highlight the difference of the algorithms, demands are increased to 1.5, 2.0 and 2.5 times of the baseline level. Figure 4 reports the convergence performance in all four scenarios. The most important finding from this experiment is the problematic performance of BOB when the demand level is high. The convergence curves of BOB and NOB are comparable for low demand levels (1.0 and 1.5). BOB was actually faster for the baseline demand (1.0). When the demand level increases to 2.0, however, BOB was trapped at the relative gap 1E-3 for many iterations until it suddenly picked up the regular pace. For the demand level of 2.5, BOB could not recover from a similar trap before the maximum number of iterations (200) was reached. In contrast, NOB’s performance is stable in all scenarios. We conjecture that the difference stemmed from the approximation of the second-order derivative, since the two algorithms differ only on that point. Apparently, the lower bound approximation in BOB is more likely to cause the wrong search direction problem (i.e. negative second order derivatives) in more congested networks. Generally higher demand levels cause more routes to be used at equilibrium. Accordingly, the interactions between different routing variables $\frac{\partial^2 z}{\partial q_i \partial q_j}$ (cf. Equation (54)) are increased, thereby shifting the lower bound approximation further from its true values.

![Figure 4 Convergence performance for the Anaheim network](image)

Besides the demand level, the convergence of BOB may be related to network topology and algorithm settings (e.g. how precisely the inner problem is solved). How these factors together affect BOB’s performance remains unclear. Nevertheless, the superiority of NOB is obvious: it not only provides a theoretically sound resolution to the “wrong direction problem”, but also outperforms BOB from a computational perspective, when the demand is high.

5.3. Experiments on other networks
The last section exhibited the unsatisfactory performance of Bar-Gera’s algorithm. This section confirms that finding on a handful of other real networks obtained from Bar-Gara’s Traffic Assignment Test Problem website (http://www.bgu.ac.il/ bargera/tntp/); see Table 2.

Table 3 reports the number of iterations and CPU times taken to reach various relative gaps, for three networks (BL, CK, WN) at two different demand levels (1.0 and 2.0). Overall BOB and NOB required similar CPU times to meet lower levels of precision. For the WN network at the demand
Table 2  Details of tested networks

<table>
<thead>
<tr>
<th>Network</th>
<th>Nodes</th>
<th>Links</th>
<th>Origins</th>
<th>OD Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barcelona (BL)</td>
<td>1020</td>
<td>2522</td>
<td>97</td>
<td>7922</td>
</tr>
<tr>
<td>ChicagoSketch (CK)</td>
<td>933</td>
<td>2950</td>
<td>386</td>
<td>142890</td>
</tr>
<tr>
<td>Winnipeg (WN)</td>
<td>1052</td>
<td>2836</td>
<td>135</td>
<td>4345</td>
</tr>
<tr>
<td>ChicagoRegional (CR)</td>
<td>12982</td>
<td>39018</td>
<td>1771</td>
<td>3136441</td>
</tr>
</tbody>
</table>

Table 3  Results for BL, CK, WN networks

<table>
<thead>
<tr>
<th>Precision level</th>
<th>Problems</th>
<th>Algorithms</th>
<th>1.0E-03</th>
<th>1.0E-06</th>
<th>1.0E-09</th>
<th>1.0E-12</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Iter</td>
<td>CPU</td>
<td>Iter</td>
<td>CPU</td>
<td>Iter</td>
</tr>
<tr>
<td>BL-1.0</td>
<td>NOB</td>
<td>3</td>
<td>10.5</td>
<td>7</td>
<td>23.4</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>BOB</td>
<td>3</td>
<td>10.4</td>
<td>7</td>
<td>23.5</td>
<td>7</td>
</tr>
<tr>
<td>BL-2.0</td>
<td>NOB</td>
<td>3</td>
<td>12.3</td>
<td>9</td>
<td>35.6</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>BOB</td>
<td>3</td>
<td>12.2</td>
<td>9</td>
<td>35.6</td>
<td>57</td>
</tr>
<tr>
<td>CK-1.0</td>
<td>NOB</td>
<td>2</td>
<td>33.3</td>
<td>6</td>
<td>96.4</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>BOB</td>
<td>2</td>
<td>34.0</td>
<td>6</td>
<td>99.8</td>
<td>12</td>
</tr>
<tr>
<td>CK-2.0</td>
<td>NOB</td>
<td>4</td>
<td>73.4</td>
<td>12</td>
<td>208.3</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>BOB</td>
<td>4</td>
<td>75.3</td>
<td>12</td>
<td>208.3</td>
<td>Best R-gap = 1.81e-005 (3528.20 sec. CPU)</td>
</tr>
<tr>
<td>WN-1.0</td>
<td>NOB</td>
<td>2</td>
<td>10.2</td>
<td>8</td>
<td>41.0</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>BOB</td>
<td>2</td>
<td>10.2</td>
<td>8</td>
<td>41.0</td>
<td>29</td>
</tr>
<tr>
<td>WN-2.0</td>
<td>NOB</td>
<td>4</td>
<td>24.1</td>
<td>16</td>
<td>94.7</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>BOB</td>
<td>4</td>
<td>24.8</td>
<td>16</td>
<td>94.7</td>
<td>Best R-gap = 6.56e-005 (1261.38 sec. CPU)</td>
</tr>
</tbody>
</table>

Note: CPU times are measured in seconds.

level 1.0, BOB outperformed NOB for all levels of precision. However, NOB beat BOB in all other scenarios, particularly for the higher demand level. In fact, BOB failed to improve the relative gap after a certain point for both CK and WN at the demand level 2.0, similar to the results from the last section. Results for the CR network (demand level = 1.0) showed a similar pattern (see Figure 5): BOB ended up with a long, flat tail while NOB maintained a roughly linear convergence rate. To achieve a relative gap of 1E-4, NOB required about 11 main iterations and 438 minutes of CPU time, while BOB took 29 main iterations and 1320 minutes of CPU time.
6. Conclusions
The “wrong direction problem” in the Bar-Gera’s algorithm (2002) for the origin-based traffic assignment problem is identified and a resolution is proposed. The problem is caused by approximating the second-order derivative with its lower bound. Our numerical experiments indicate that this problem negatively impacts the convergence performance of Bar-Gera’s algorithm, particularly when the network is congested. In the worst case, the algorithm may become stuck well before satisfactory convergence is achieved.

We prove that the wrong direction can be completely avoided if Bertsekas’s upper bound (1984) is used to approximate the second-order derivative. This alternative not only fixes the theoretical flaw, but indeed demonstrates more desirable computational performance, as evidenced by our numerical results. Moreover, upper bounds can be computed almost as efficiently as lower bounds, in a similar recursive manner.

With the improvement demonstrated in this note, the relative performance of origin-based algorithms warrants a thorough investigation to generate useful insights to the search of the next “best” TAP-algorithm[4]. Such an effort would necessarily include a careful comparison of various sub-problem solvers and strategies used to expand/trim acyclic networks. Indeed, our experiments indicate that the performance of origin-based algorithms is substantially affected by these factors.

Acknowledgments
The author would like to thank Professor David Boyce for discussions that have inspired this work and his comments on various drafts of this paper. The author is very grateful to Dr. Hillel Bar-Gera for sharing his OBA source code in 2000, through Professor Der-Horng Lee at National University of Singapore. Dr. Bar-Gera’s constructive comments have greatly improved the derivation presented in Section 3. Comments of two anonymous referees, as well as advice from the associate editor, are appreciated. Professor Hani Mahmassani, the immediate past editor-in-chief of the journal, also offered generous assistance when I revised an earlier version of the paper. The views expressed in this paper are the author’s alone.

References

4 While the present paper was under review, a follow-up study along this direction had been completed and subsequently published in Nie (2010).


Appendix. List of Notation

Network:
- $A$: set of links
- $N$: set of nodes
- $n$: number of nodes
- $R$: set of origin nodes
- $S$: set of destination nodes
- $K_{rj}$: set of simple routes between $r$ and $j$
- $B_r$: bush associated with origin $r$
- $O(j)$: set of successor(s) of node $j$
- $I(j)$: set of predecessor(s) of node $j$
- $\pi_j$: topological distance of node $j$ with respect to the origin
- $\chi(j)$: last common node of node $j$

Flows:
- $x_{ij}^r$: flow on link $ij$ contributed by all O-D pairs $rs$ originating at $r$
- $x_{ij}'$: flow on link $ij$ contributed by all O-D pairs $rs$ not originating at $r$
- $x_{ij}$: total flow on link $ij$
- $q_j'$: flow terminating at node $j$ from the origin $r$ (given)
- $q_j$': same as $q_j'$, the index $r$ is suppressed due to the single-origin formulation
- $\eta_j$: flow arriving at node $j$ from the origin
- $\phi_{ij}$: routing variable, proportion of flows arriving at node $j$ through predecessor $i$
- $\phi_{ij}$: reference routing variable for the constrained formulation
- $p_{km}'$: proportion of traffic arriving node $j$ through link $km$.
- $d_{rs}$: travel demand departing from origin $r$ to destination $s$
- $f_k'$: flow on route $k \in K_{rj}$

Cost:
- $t_{ij}$: cost on link $ij$, a strictly positive and increasing function of $x_{ij}$
- $t_{ij}'$: derivative of link cost $t_{ij}$
- $\bar{c}_{ij}':$ cost on route $k \in K_{rj}$
- $v_{ij}$: average cost from the origin to node $j$ through node $i$
- $\bar{v}_j$: average cost from the origin to node $j$
- $s_{ij}$: approximated derivative of $\bar{c}_{ij}$ with respect to $x_{ij}$
- $\hat{s}_j$: approximated derivative of $\bar{c}_j$ with respect to $\eta_j$
- $u_{ij}'$: minimum cost from the origin $r$ to node $j$
- $u_j$: same as $u_j'$, the index $r$ is suppressed due to the single-origin formulation
- $\mu_j$: Lagrangian multiplier associated non-negative constraints for $\phi_{ij}$

Others:
- $z$: objective function
- $z_c$: objective function defined on the space of non-reference routing variables
- $\Phi$: $n \times n$ matrix that relates $q_j$ to $\eta_j, \forall j$ through routing variables
- $\Lambda$: inverse of $\Phi$
- $\lambda_i$: entry in $\Lambda$, proportion of traffic arriving $j$ through $i$
- $\delta_{jk}$: an incidence variable, equals 1 if $j = k$ and 0 otherwise
- $\eta'$: vector of derivatives of $\eta_j, \forall j$ with respect to $\phi_{km}$ for any fixed $km$
- $\hat{\eta}$: vector of $\eta_j \delta_{jk}, \forall j$ for any fixed $k$
- $\alpha$: step size
- $g_r$: relative gap