A superlinearly convergent Newton-like method for linearly constrained optimization problems is adapted for solution of multicommodity network flow problems of the type arising in communication and transportation networks. We show that the method can be implemented approximately by making use of conjugate gradient iterations without the need to compute explicitly the Hessian matrix. Preliminary computational results suggest that this type of method is capable of yielding highly accurate solutions of nonlinear multicommodity flow problems far more efficiently than any of the methods available at present.
PROJECTED NEWTON METHODS AND OPTIMIZATION OF MULTICOMMODITY FLOWS*

by

Dimitri P. Bertsekas and Eli M. Gafni
Department of Electrical Engineering and Computer Science
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

Abstract

A superlinearly convergent Newton-like method for linearly constrained optimization problems is adapted for solution of multicommodity network flow problems of the type arising in communication and transportation networks. We show that the method can be implemented approximately by making use of conjugate gradient iterations without the need to compute explicitly the Hessian matrix. Preliminary computational results suggest that this type of method is capable of yielding highly accurate solutions of nonlinear multicommodity flow problems far more efficiently than any of the methods available at present.

*This research was conducted at the M.I.T. Laboratory for Information and Decision Systems with partial support provided by the National Science Foundation Grant ECS-79-20834 and the Defense Advanced Research Project Agency Grant ONR-N00014-75-C-1183.
1. Introduction

The methods that are currently most popular for solving smooth linearly constrained optimization problems of the form

$$\begin{align*}
\text{minimize} & \quad J(x) \\
\text{subject to} & \quad Ax \leq b,
\end{align*}$$

where $J: \mathbb{R}^n \to \mathbb{R}$, $A: \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$, are based on solution of some type of linear or quadratic programming subproblems. For example, methods stemming from the original proposals of Goldstein [1], and Levitin and Poljak [2] take the form

$$x_{k+1} = x_k + \alpha_k (\bar{x}_k - x_k)$$

where $\bar{x}_k$ solves

$$\begin{align*}
\text{minimize} & \quad \nabla J(x_k)'(x-x_k) + \frac{1}{2} (x-x_k)'H_k(x-x_k) \\
\text{subject to} & \quad Ax \leq b,
\end{align*}$$

$H_k$ is a positive definite matrix, and $\alpha_k$ is a positive scalar stepsize determined according to some rule. This method is capable of superlinear convergence if $H_k$ is either the Hessian matrix $\nabla^2 J$ or some suitable Quasi-Newton approximation of $\nabla^2 J$ [2]-[4]. However, for large-dimensional problems the overhead for solving problem (3) is typically prohibitive with such a choice of $H_k$ thereby rendering the method impractical.

The difficulty with excessive overhead in solving the quadratic programming problem (3) can be bypassed in at least two ways if the constraint set has a simple form (for instance upper and lower bounds on the coordinates of
x, Cartesian products of simplices, etc.), or has special structure (for example it expresses conservation of flow equations for the nodes of a directed graph). One possibility is to take $H_k = 0$ in problem (3) so that (3) becomes a linear program. This leads to methods of the Frank-Wolfe type [5] which has been extensively applied for solution of multicommodity network flow problems [6],[8]. The rate of convergence of these methods is sublinear [9], [10] and therefore too slow for applications where high solution accuracy is demanded. The other possibility is to choose the matrix $H_k$ in (3) to be positive definite and diagonal. With such a choice it is often possible to solve the quadratic subproblem (3) very efficiently by exploiting the simple structure of the constraint set. Methods of this type have a long and quite successful history in large-scale problems arising in network flow applications [7], [11]-[17] as well as in other areas such as optimal control [18], [19]. However their rate of convergence is typically linear and in many applications unacceptably slow.

A somewhat different type of method stems from the original gradient projection proposal of Rosen [20], and other related proposals (the reduced gradient method and the convex simplex method [21] etc.). The typical iteration in these methods proceeds along a linear manifold of active constraints which is gradually modified during the algorithm as previously active constraints become inactive and new constraints become active (see [22]-[25]). These methods are quite effective for problems of small dimension and have also been applied in some network flow problems [26], [27], but, in our view, are highly unsuitable for large problems with many constraints. The main reason is that they typically allow only one new constraint to become active in any one iteration. So if for example one
thousand constraints are active at the solution which are not active at the starting point, these methods require at least one thousand iterations and likely many more in order to converge.

In this paper we consider a projected Newton method first proposed in Bertsekas [28] that offers substantial and often decisive advantages over the methods described above for large problems with many simple constraints as typified by a multicommodity flow structure. For the problem

\[
\text{minimize} \quad J(x) \\
\text{subject to} \quad x > 0
\]

it takes the simple form

\[
x_{k+1} = [x_k - \alpha_k D_k \nabla J(x_k)]^+ \tag{5}
\]

where \( \alpha_k \) is a positive scalar stepsize, \( D_k \) is a positive definite symmetric matrix which is diagonal with respect to some of the coordinates of \( x \), and \([\cdot]^+\) denotes projection (with respect to the standard norm) on the positive orthant. It is shown in [28] that \( D_k \) can be chosen on the basis of second derivatives of \( J \) so that the method typically converges superlinearly.

Iteration (5) constitutes the basic building block for extensions to more general inequality constrained problems by means of a procedure described in [28]. In this paper we focus on the case where the constraint set is a Cartesian product of simplices, and consider in more detail a class of nonlinear multicommodity flow problems characterized by a constraint set of this type. We describe an approximate version of a Newton-like method based on approximate solution of the Newton system of equations via the
conjugate gradient method. It turns out that for network flow problems this conjugate gradient method can be implemented very efficiently by network type operations—a fact also observed earlier in a different context by Dembo [29]. As a result a significant advantage in speed of convergence is gained over earlier methods at the expense of relatively small additional overhead per iteration.

The notation employed throughout the paper is as follows. All vectors are considered to be column vectors. A prime denotes transposition. The standard norm in $\mathbb{R}^n$ is denoted by $|\cdot|$, i.e. for $x = (x^1, \ldots, x^n)$ we write $|x| = \left[ \sum_{i=1}^{n} (x^i)^2 \right]^{1/2}$. The gradient and Hessian of a function $f: \mathbb{R}^n \to \mathbb{R}$ are denoted by $\nabla f$ and $\nabla^2 f$ respectively. All vector inequalities are meant to be componentwise (for example $x \geq 0$ means $x^i \geq 0$, $i = 1, \ldots, n$).

2. A Projected Newton Method for Minimizing a Twice Differentiable Function on a Simplex

Consider the problem

$$\begin{align*}
\text{minimize} & \quad J(x) \\
\text{subject to} & \quad x \geq 0, \quad \sum_{i=1}^{n} x^i = r
\end{align*}$$

(6)

where $J: \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and $r$ is a given positive scalar. We also assume for convenience that $J$ is convex although generalizations of all the results and algorithms of this paper are possible without this assumption.

We describe the $k$th iteration of a Newton-like method for solving (6). At the beginning of the iteration we have a feasible vector $x_k$. The next (feasible) vector $x_{k+1}$ is obtained by means of the following procedure:
By rearranging indices if necessary assume that the last coordinate \( x^n \) satisfies
\[
x^n = \max \{ x_i^k | i = 1, \ldots, n \}.
\]

Consider a reduced coordinate system in the vector \( y \in \mathbb{R}^{n-1} \) given by
\[
y = (y^1, \ldots, y^{n-1}) = (x^1, x^2, \ldots, x^{n-1}),
\]
denote \( y_k = (x^1_k, \ldots, x^{n-1}_k) \), and consider the reduced objective function
\[
h_k(y) = J(y^1, \ldots, y^{n-1}, r - \sum_{i=1}^{n-1} y^i).
\]

Based on this transformation problem (6) is equivalent locally (around \( y_k \)) to the problem

\[
\text{minimize } h_k(y) \quad y \geq 0
\]
in the sense that the constraint \( r - \sum_{i=1}^{n-1} y^i \geq 0 \) is (by construction) inactive within a neighborhood of \( y_k \). The following iteration is based on this fact [compare with (4),(5)]. For an \( (n-1) \times (n-1) \) positive definite symmetric matrix \( D_k \) to be further specified later denote
\[
y_k(\alpha) = [y_k - \alpha D_k V h_k(y_k)]^+, \quad \forall \alpha \geq 0
\]
where \([\cdot]^+\) denotes projection on the positive orthant [i.e. for a vector \( y = (y^1, \ldots, y^{n-1}) \), the vector \([y]^+\) has coordinates \( \max\{0, y^i\}, i = 1, \ldots, n-1\)].

Define the vector \( y_{k+1} \) by means of
\[
y_{k+1} = y_k(\alpha_k)
\]
where the stepsize $\alpha_k$ is chosen by means of a rule to be specified further later from the range

$$\alpha_k \in [0, \alpha_k^\ast]$$

(13)

with $\alpha_k^\ast$ given by

$$\alpha_k^\ast = \sup \{a \mid \sum_{i=1}^{n-1} y_k^i(a) \leq r\}.$$  

(14)

[Note that in view of (7), (8), (11), we have $\alpha_k^\ast > 0$ or $\alpha_k^\ast = \infty$.]

The next vector $x_{k+1}$ generated by the algorithm has coordinates given by

$$x_{k+1}^i = y_{k+1}^i, \quad i = 1, \ldots, n$$

(15a)

$$x_{k+1}^n = r - \sum_{i=1}^{n-1} y_{k+1}^i$$

(15b)

We first note that, in view of (11), (13), (14) the vector $x_{k+1}$ is feasible. The following proposition identifies a class of matrices $D_k$ for which a descent iteration is obtained.

Denote

$$I_k^+(x_k) = \{i \mid y_k^i = 0, \frac{\partial h_k(y_k)}{\partial y^i} > 0\}$$

(16)

and consider for all $\alpha \geq 0$ the vector $x_k(\alpha)$ with coordinates given by

$$x_k^i(\alpha) = y_k^i(\alpha), \quad i = 1, \ldots, n-1$$

(17)

$$x_k^n(\alpha) = r - \sum_{i=1}^{n-1} y_k^i(\alpha).$$

(18)
Proposition 1: Assume that the positive definite symmetric matrix $D_k$ is diagonal with respect to the index set $I^+_k(x_k)$ in the sense that the elements $D_{ij}^{k}$ of $D_k$ satisfy

$$D_{ij}^{k} = 0$$

for all $i \in I^+_k(x_k)$ and $j = 1, \ldots, n$ with $i \neq j$.

a) If $x_k$ is a global minimum of problem (6) then

$$x_k(\alpha) = x_k, \quad \forall \alpha \geq 0$$

b) If $x_k$ is not a global minimum of problem (6) then there exists $\tilde{\alpha} \in (0, \bar{\alpha}_k]$ such that for all $\alpha \in (0, \tilde{\alpha}]$ the vector $x_k(\alpha)$ is feasible, and

$$f[x_k(\alpha)] < f(x_k), \quad \forall \alpha \in (0, \tilde{\alpha}]. \quad (19)$$

The proposition above shows that the algorithm essentially terminates at a global minimum and is capable of descent when not at a global minimum.

There are a number of issues relating to selection of the matrix $D_k$ and the stepsize $\alpha_k$ and associated questions of convergence and rate of convergence which are addressed in [28] and will only be summarized here. We first mention that the convergence results available require that $D_k$ is not only diagonal with respect to the set $I^+_k(x_k)$ but rather with respect to the possibly larger set

$$I^+_k = \{i \mid 0 \leq y^i_k \leq \epsilon^i_k, \quad \frac{\partial h_k(y_k)}{\partial y^i_k} > 0\} \quad (20)$$

where

$$\epsilon^i_k = \min\{\epsilon, s^i_k\} \quad (21)$$
\( c \) is a fixed positive scalar, \( s_k^i \) is given by
\[
s_k^i = \left| y_k^i - \left( y_k^i - \mu_k^i \frac{\partial h_k(y_k)}{\partial y^i} \right)^+ \right|
\] (22)
and \( \mu_k^i \) are scalar sequences such that
\[
\mu_k^i \geq \mu^i > 0, \quad k = 0, 1, \ldots
\]
with \( \mu^i \) being some positive scalars which are fixed throughout the algorithm.

This is an antizagzagging device of the type commonly employed in feasible direction methods (see e.g. [30]), and is designed to counteract the possible discontinuity exhibited by the set \( I_k^+(x_k) \) as \( x_k \) approaches the boundary of the positive orthant. (Actually the formula used in [28] is slightly different than (22) but this difference does not affect the convergence and rate of convergence results of [28]).

Regarding the choice of the stepsize \( \alpha_k \), there are at least two practical methods that lead to algorithms which are demonstrably convergent. In the first method \( \alpha_k \) is chosen according to
\[
\alpha_k = \min\{\bar{\alpha}, \frac{\bar{\alpha}_k}{\bar{\alpha}}\}
\] (23)
where \( \bar{\alpha} \) is a fixed positive constant and \( \bar{\alpha}_k \) is given by (14). In the second method an initial stepsize is chosen and is successively reduced by a certain factor until a "sufficient" reduction (according to an Armijo-like test) of the objective function is observed [28]. Under further mild assumptions it is possible to show that all limit points of sequences generated by the algorithm are global minima of problem (6). Furthermore
after some index the sets $I_k^+$ are equal to both $I_k^+(x_k)$ and the set of indices of coordinates of $y_k$ that are zero at the limit point. This last property is instrumental in constructing superlinearly convergent algorithms as it shows that the portion of the matrix $D_k$ which must be "diagonalized" plays no role near the end of the algorithm. As a result superlinear convergence can be achieved by choosing the portion of the matrix $D_k$ that corresponds to the indices not in $I_k^+$ to be equal to the inverse Hessian of $h_k$ with respect to these indices. The $k$th iteration of the resulting algorithm can be restated as follows:

First the set $I_k^+$ is calculated according to (20)-(22) on the basis of the gradient $\nabla h_k$. Then the vector $y$ is partitioned as in

$$y = \begin{bmatrix} \tilde{y} \\ \bar{y} \end{bmatrix}$$

(24)

where $\tilde{y}$ is the vector of coordinates $y_i$ with $i \in I_k^+$ and $\bar{y}$ is the vector of coordinates $y_i$ with $i \in I_k^-$. Then a "search direction" $d_k = (\tilde{d}_k, \bar{d}_k)$ is obtained solving the systems of equations

$$\tilde{H}_k \tilde{d} = -\tilde{g}_k$$

(25)

$$\bar{H}_k \bar{d} = -\bar{g}_k$$

(26)

where $\tilde{g}_k$ (or $\bar{g}_k$) is the vector with coordinates $\frac{\partial h_k(y_k)}{\partial y_i}$ with $i \in I_k^+$ (respectively $i \in I_k^-$), $\tilde{H}_k$ is a diagonal positive definite matrix, and $\bar{H}_k$ is a symmetric positive definite matrix which is equal to the Hessian of $h_k$ with respect to the coordinates $y_i, i \in I_k^+$. The vector $y_{k+1}$ is then obtained by
where $\alpha_k$ is the stepsize obtained according to one of the rules mentioned earlier.

We wish to call the reader's attention to the natural decomposition of the iteration into three phases: The formation of the index set $I^*_k$, the computation of the "search direction" $d_k$, and the determination of the stepsize $\alpha_k$. There is considerable freedom for variations in each phase independently of what is done in other phases while still maintaining desirable convergence and rate of convergence properties.

**Approximate Implementation via the Conjugate Gradient Method**

Finding the "search direction" $d_k$ requires the solution of the linear system of equations (26). Solution of this system can be accomplished, of course, by a finite method involving triangular factorization but when the dimension of this system is large, as for example in multicommodity flow problems, the associated computational overhead can make the overall algorithm impractical. The alternative is to solve this system iteratively by, for example, a successive overrelaxation method or a conjugate gradient method. This approach is practiced widely by numerical analysts [31] and its success hinges upon the ability of the iterative method to yield a good approximation of the solution of system (26) within a few iterations. In order to guarantee convergence of the overall optimization algorithm it is necessary that the approximate solution, call it $\bar{z}$, of the system (26) satisfies

$$\bar{z}^T g_k < 0$$

(28)
whenever $\bar{g}_k \neq 0$, in order to make possible a reduction in the objective function value [cf. Proposition 1b)]. This is the minimal requirement that we impose upon the iterative method used to solve (26).

In this paper we are primarily interested in approximate solution of the system

$$\bar{H}_k z = -\bar{g}_k,$$

(29)

or equivalently the unconstrained minimization problem

$$\min_z \bar{g}_k^T z + \frac{1}{2} z^T \bar{H}_k z$$

(30)

by means of the following scaled version of the conjugate gradient method:

A positive definite symmetric matrix $S_k$ is chosen, and a sequence \( \{z_m\} \) is generated according to the iteration

$$z_0 = 0, \quad z_{m+1} = z_m + \gamma_m p_m, \quad m = 0,1,\ldots$$

(31)

where the conjugate direction sequence \( \{p_m\} \) is given recursively by

$$p_0 = -S_k r_0, \quad p_m = -S_k r_m + \beta_m p_{m-1}, \quad m = 1,2,\ldots$$

(32)

the residual sequence \( \{r_m\} \) is defined by

$$r_m = \bar{H}_k z_m + \bar{g}_k, \quad m = 0,1,\ldots$$

(33)

and the scalars $\gamma_m$ and $\beta_k$ are given by

$$\gamma_m = \frac{r_m^T S_k r_m}{p_m^T \bar{H}_k p_m}, \quad m = 0,1,\ldots$$

(34)
\[
\beta_m = \frac{r^t S_m r_{m+1}}{r^t S_k r_{m-1}}, \quad m = 1, 2, \ldots \tag{35}
\]

As is well known ([25], [32]) this method will find the solution \( \bar{d}_k \) of system (29) in at most \( (n-1) \) steps (i.e., \( \bar{d}_k = z_{n-1} \)) regardless of the choice of \( S_k \). We are primarily interested however in approximate implementations whereby only a few conjugate gradient iterations of the method are performed and under these circumstances the choice of \( S_k \) can have a substantial effect on the quality of the final approximate solution. A suitable choice for many problems (and the one we prefer for multicommodity flow problems) is to choose \( S_k \) to be diagonal with elements along the diagonal equal to the second derivatives of the \( h_k \) with respect to the corresponding coordinates \( y^i, i \in I_k \) evaluated at \( y_k \). There are however other attractive possibilities depending on problem structure (see [33]).

It is easily verified that if \( g_k \neq 0 \), then we have

\[
z^t_m g_k < 0, \quad \forall \ m = 1, 2, \ldots
\]

so, regardless of how many conjugate gradient iterations are performed, the final approximate solution \( z \) of system (29) will satisfy the descent condition (28).

We finally mention that the assumption that \( H_k \) be positive definite is not strictly necessary for the preceding algorithm to generate a descent direction. It is sufficient that \( g_k \neq 0 \) and \( H_k \) be such that the quadratic optimization problem (30) have at least one globally optimal solution. It turns out that this minor refinement is significant for the multicommodity flow problems to be considered in the next section.
Extension to the Case where the Constraint Set is a Cartesian Product of Simplices

Consider the problem

\[
\text{minimize } J[x(1), \ldots, x(m)] \quad (37)
\]
subject to \( x(j) \geq 0 \), \( \sum_{i=1}^{n_j} x_i(j) = r_j, j = 1, \ldots, m \)

where each \( x(j), j = 1, \ldots, m \) is a vector in \( \mathbb{R}^{n_j} \), the function \( J: \mathbb{R}^{n_1 + \ldots + n_m} \rightarrow \mathbb{R} \) is twice continuously differentiable and \( r_j, j = 1, \ldots, m \) are given positive scalars.

The extension of the method described earlier in this section to handle problem (37) is evident once it is realized that one can similarly pass to a reduced coordinate system of dimension \( (n_1 + \ldots + n_m - m) \) while in the process eliminating the \( m \) equality constraints \( \sum_{i=1}^{n_j} x_i(j) = r(j), j = 1, \ldots, m \), [cf. (8), (15)]. One then obtains a reduced problem involving nonnegativity constraints only [cf. (9), (10)] which is locally (around the current iterate) equivalent to problem (37). The iteration described earlier, including the conjugate gradient approximation process, is fully applicable to the reduced problem.

3. Optimization of Multicommodity Flows

We consider a network consisting of \( N \) nodes \( 1, 2, \ldots, N \) and a set of directed links denoted by \( L \). We denote by \((i, k)\) the link from node \( i \) to node \( k \), and assume that the network is connected in the sense that for any two nodes \( m, n \) there is a directed path from \( m \) to \( n \). We are given a set \( W \) of ordered node pairs referred to as origin-destination (or OD) pairs. For
each OD pair \( w \in W \), we are given a set of directed paths \( P_w \) that begin at the origin node and terminate at the destination node. For each \( w \in W \) we are also given a positive scalar \( r_w \) referred to as the input of OD pair \( w \), and this input must be optimally divided among the paths in \( P_w \) so as to minimize a certain objective function.

For every path \( p \in P_w \) corresponding to an OD pair \( w \in W \), we denote by \( x^p \) the flow travelling on \( p \). These flows must satisfy

\[
\sum_{p \in P_w} x^p = r_w, \quad \forall w \in W \tag{38}
\]

\[
x^p > 0, \quad \forall p \in P_w, w \in W. \tag{39}
\]

Equations (38), (39) define the constraint set of the optimization problem—a Cartesian product of simplices.

To every set of path flows \( \{x^p \mid p \in P_w, w \in W\} \) satisfying (38), (39) there corresponds a flow \( f_{i \ell} \) for every link \((i, \ell)\). It is defined by the relation

\[
f_{i \ell} = \sum_{w \in W} \sum_{p \in P_w} \delta_p(i, \ell) x^p, \quad \forall (i, \ell) \in L \tag{40}
\]

where \( \delta_p(i, \ell) = 1 \) if the path \( p \) contains the link \((i, \ell)\) and \( \delta_p(i, \ell) = 0 \) otherwise. If we denote by \( x \) and \( f \) the vectors of path flows and link flows respectively we can write relation (40) as

\[
f = Ex \tag{41}
\]

where \( E \) is the arc-chain matrix of the network.

For each link \((i, \ell)\) we are given a convex twice continuously differentiable scalar function \( D_{i \ell}(f_{i \ell}) \) with strictly positive second derivative.
for all \( f_{ij} \geq 0 \). The objective function is given by

\[
D(f) \triangleq \sum_{(i,j) \in L} D_{ij}(f_{ij}).
\]

(42)

By using (41) we can write the problem in terms of the path flow variables \( x_P \) as

\[
\text{minimize } J(x) \triangleq D(Ex)
\]

subject to \( \sum_{p \in P_w} x_P = r_w, \forall w \in W \)

\[
x_P \geq 0, \forall p \in P_w, w \in W.
\]

In communication network applications the function \( D \) may express, for example, average delay per message [6], [11] or a flow control objective [34], while in transportation networks it may arise via a user or system optimization principle formulation [16], [17], [35]. The algorithm to be presented admits an extension to the case where the function \( D \) does not have the separable form (42), but we prefer to concentrate on the simpler and practically important separable case in order to avoid further complications in our notation.

Clearly problem (42) falls within the framework of the previous section and the approximate version of the projected Newton method described there can be applied for its solution. A key element for the success of this algorithm lies in that the conjugate gradient iterations required for approximate solution of the corresponding system of equations can be carried out very efficiently. This in turn hinges on the fact that the product of

\[
\text{product of}
\]
the matrix $H_k$ with various vectors, which is needed for the computation of
the residual $r_m$ in (33) and the stepsize $\gamma_m$ in (34), can be computed by
graph type operations without explicitly computing or storing the matrix $H_k$.

We now describe the $k$th iteration of the algorithm whereby given a
feasible vector of path flows $x_k$ we find the next vector $x_{k+1}$:

**Phase 1:** (Determination of the reduced coordinate system and the set $I_k^+$).

For each $w \in W$ let $p_w$ be the path carrying maximal flow, i.e.,

$$P_w = \{i, I, \ldots, J, w\}, \quad V \subseteq W$$

(44)

Define the reduced coordinate system in the vector $y$ given by [cf. (8)]

$$y^P = x^P, \quad \forall p \in P, \quad w \in W,$$

(45)

and denote by $y_k$ the vector that corresponds to $x_k$ according to this trans-
formation. Consider the reduced objective function $h_k(y) = J(x)$ [cf. (9)]
where $x$ has coordinates given by $x^P = y^P, \forall p \in P$ and $w \in W$ and

$$x^P = r_w - \sum_{p \in P, w} x^P,$$

(46)

Denote $D'_{ij}$ and $D''_{ij}$ the first and second derivatives of $D_{ij}$ evaluated
at $x_k$, and define the first derivative length of a path $p$ by

$$l_p = \sum_{(i, j) \in p} D'_{ij}, \quad \forall p \in P, \quad w \in W,$$

(47)

i.e., $l_p$ is the sum of first derivatives $D'_{ij}$ over all links on the path $p$.

It is easily verified that
and that the gradient of the reduced objective function is given by
\[
\frac{\partial h_k(y_k)}{\partial y^p} = l_p - l_{p_w}, \quad \forall \, p \in P_w, \, w \in W
\]
(49)

By differentiating this expression with respect to \( y^p \) we also find after a straightforward calculation the diagonal elements of the Hessian \( \nabla^2 h_k \)
\[
\frac{\partial^2 h_k(y_k)}{(\partial y^p)^2} = \sum_{(i, k) \in L_p} D''_{i_k}, \quad \forall \, p \in P_w, \, p \neq p_w, \, w \in W
\]
(50)

where \( L_p \) is the set of links that are traversed by either the path \( p \) or the path \( p_w \) but not both. In view of our assumption \( D''_{i_k} > 0 \) for all \( f_{i_k} \geq 0 \) we see that there exist scalars \( \mu^p \) such that
\[
\mu^p_k = \left[ \frac{\partial^2 h_k(y_k)}{(\partial y^p)^2} \right]^{-1} > \mu^p > 0, \quad \forall \, p \in P_w, \, p \neq p_w, \, w \in W
\]
(51)

for all feasible vectors \( y_k \).

We are now in a position to define the set \( I^+_k \) in terms of a positive scalar \( \varepsilon > 0 \) which remains fixed throughout the algorithm. We set [cf. (20)-(22), (49)-(51)]
\[
I^+_k = \{ p \mid 0 \leq y^p_k \leq \varepsilon^p_k, \, \mu^p > 1, \, p \in P_w, \, p \neq p_w, \, w \in W \}
\]
(52)

where
\[
\varepsilon^p_k = \min \{ \varepsilon, \varepsilon^p_k \}
\]
(53)
and
An equivalent definition is that a path \( p \) belongs to \( I_k^+ \) if it has a larger first derivative length than the corresponding reference path \( p_w \), and it carries flow that is less or equal to both \( \varepsilon \) and \( \frac{i_k}{k}(l_p - l_{p_w}) \). As will be seen later the algorithm "tries" to set the flow of these paths to zero [cf. (57), (69)].

**Phase 2: (Computation of the search direction)**

As in the previous section we form a partition of the vector \( y \) [cf. (24)]

\[
y = \begin{bmatrix} \tilde{y} \\ \check{y} \end{bmatrix}
\]

(55)

where \( \tilde{y} \) is the vector of path flows \( y_p^p \) with \( p \in I_k^+ \) and \( \check{y} \) is the vector of path flows \( y_p^p \) with \( p \not\in I_k^+ \). The search direction \( d_k \), partitioned consistently with (55)

\[
d_k = \begin{bmatrix} \check{d}_k \\ \tilde{d}_k \end{bmatrix}
\]

(56)

is defined as follows [cf. (25), (26)]. For paths \( p \in I_k^+ \) we set

\[
\check{d}_k^p = -\frac{i_k}{k}(l_{p_b} - l_{p_w}), \quad \forall p \in I_k^+
\]

(57)

i.e. the matrix \( H_k \) of (25) is set to the diagonal matrix with elements

\[
\frac{\partial^2 h_k(y_k)}{(2y_p^p)^2} \quad \text{along the diagonal.}
\]

For paths \( p \not\in I_k^+ \) the search direction is defined by

\[
\bar{H}_k \bar{d}_k = -\bar{g}_k
\]

(58)
where \( \overline{g}_k \) is the gradient of \( h_k \) with respect to \( \overline{y} \) having coordinates 
\[
(1_{p-1}, p \overline{p}_w^v), \text{ [cf. (49)]}
\]
and \( \overline{H}_k \) is the Hessian matrix of \( h_k \) with respect to \( \overline{y} \).

This equation will be solved (perhaps approximately) by means of the conjugate gradient method described in the previous section [cf. equations (30)–(35)]. As scaling matrix \( S_k \) in (32) and (35) we will choose the diagonal matrix with diagonal elements the scalars \( \mu_k^p, p \notin I_k, p \neq p_w, \text{ weW}, \)
given by (50) and (51). From equations (31)-(35) it is evident that the only difficult part in implementing the conjugate gradient iteration lies in computing vectors of the form

\[
v = \overline{H}_k \Delta y \tag{59}
\]

where \( \Delta y \) is any vector of dimension equal to the number of paths \( p \) with \( p \notin I_k \) and \( p \neq p_w, \text{ weW} \). There are two such vectors to be computed at each iteration, the vector \( \overline{H}_k z_m \) appearing in the residual equation

\[
r_m = \overline{H}_k z_m + \overline{g}_k
\]

[cf. (33)], and the vector \( \overline{H}_k p_m \) appearing in the stepsize equation

\[
\gamma_m = \frac{r_m^T S_k r_m}{\overline{p}_m^T \overline{H}_k p_m} \tag{50}
\]

[cf. (34)]. However it is important to note that only one of these vectors (the vector \( \overline{H}_k p_m \)) needs to be computed by solution of an equation such as (59) at each conjugate gradient iteration.

Indeed this iteration has the form [cf (31)]

\[
z_0 = 0, \quad z_{m+1} = z_m + \gamma_m p_m, \quad m = 0, 1, \ldots
\]
and therefore we have

\[ H_{k} z_{0} = 0, \quad H_{k} z_{m+1} = H_{k} z_{m} + \gamma_{m} H_{k} p_{m}, \quad m = 0, 1, \ldots \]

Hence the vector \( H_{k} z_{m+1} \) can be computed from the previous vector \( H_{k} z_{m} \) and the vector \( H_{k} p_{m} \).

A key fact is that in order to compute, for a given \( \Delta y \), the vector \( v = H_{k} y \) of (59) we need not form explicitly the matrix \( H_{k} \) and multiply it with \( \Delta y \). Indeed consider the following function

\[ G_{k}(\Delta f) = \frac{1}{2} \sum_{(i, j) \in L} (\Delta f_{i,j})^{2} D_{i,j} \]  \hspace{1cm} (60)

of the incremental flow vector \( \Delta f \) and the corresponding function of the reduced incremental path flow vector \( \Delta y \)

\[ M_{k}(\Delta y) = G_{k}(E \Delta x) \]  \hspace{1cm} (61)

obtained via the transformation

\[ \Delta f = E \Delta x \]  \hspace{1cm} (62)

[cf. (41)] and the transformation

\[ \Delta y_{p} = \Delta x_{p}, \quad \forall p \in P_{w}, p \in I^{+}, p \neq p_{w}, \ w \in W, \]  \hspace{1cm} (63)

\[ \Delta x_{p} = 0, \quad \forall p \in I^{+} \]  \hspace{1cm} (64)

\[ \Delta x_{w} = - \sum_{p \in P_{w}, p \neq p_{w}} y_{p}^{p}, \quad \forall w \in W. \]  \hspace{1cm} (65)
The Hessian of the function $G_k$ is the same as the Hessian of the objective function $D$ evaluated at the flow vector $f^k$ corresponding to $x^k$, and consequently the Hessian of the function $M_k$ with respect to the vector $y$ is equal to the matrix $\overline{H}_k$. For any vector $\Delta y$ the vector $v = \overline{H}_k \Delta y$ is therefore equal to

$$v = \overline{H}_k \Delta y = \nabla M_k(\Delta y). \tag{66}$$

On the other hand we have already shown how to compute the gradient of functions such as $M_k$ [cf. (47)-(49)]. The procedure consists of finding the incremental flow vectors $\Delta f_{i\ell}$ corresponding to $\Delta y$ according to (62)-(64) and (65) and forming the products $D''_{i\ell} \Delta f_{i\ell}$ for each link. Then the coordinates of the vector $v$ of (66) are given by [cf. (48), (49)].

$$v^p = \sum_{(i,\ell) \in P} D''_{i\ell} f_{i\ell} - \sum_{(i,\ell) \in P_w} D''_{i\ell} f_{i\ell}$$

$$\forall p \in P, \ p \notin I_k^+ \ p \neq P_w, \ w \in W. \tag{67}$$

Thus the products $\overline{H}_{k,m}$ and $\overline{H}_{k,m}^p$ appearing in the basic iteration of the conjugate gradient method (31)-(35) can be calculated by the procedure described above without the need to compute or store the matrix $\overline{H}_k$. Since all other operations in (31)-(35) require either the formation of inner products of vectors or the multiplication of a vector with a diagonal matrix it can be seen that the Newton-like method can be implemented via the conjugate gradient method by graph-type operations and without explicit computation or storage of any Hessian matrix.

In a practical implementation of the algorithm one should not try to solve the system (58) exactly at each iteration since this typically
requires a large number of iterations of the conjugate gradient method. Rather one should terminate the conjugate gradient iterations according to some criterion. Some possible criteria are as follows:
a) Terminate after a fixed number of conjugate gradient iterations.
b) Terminate at an iteration \( m \) if the residual \( r_m \) satisfies

\[
|r_m| \leq \beta_k |r_0|
\]  

(68)

where \( \beta_k \) is some scalar factor less than unity which may depend on the iteration index \( k \).

Taking \( \beta_k = 0 \) means solving the system \( \bar{\bar{H}}_k \Delta \bar{Y}_k = -\bar{g}_k \) exactly and yields Newton's method. Thus if \( \beta_k \to 0 \) one expects that it is possible to construct a method that realizes the superlinear convergence rate of Newton's method by making use of a proper method for choosing the stepsize \( \alpha_k \). (A result of this type is shown for the unconstrained Newton's method in [36]).

Phase 3: (Determination of the stepsize \( \alpha_k \))

As usual in Newton-like methods, we first try a unity stepsize and subsequently reduce it if certain conditions are not satisfied. Thus we form the vector

\[
\hat{y}_{k+1} = [y_k + d_k]^+
\]

(69)

where \( d_k \) is the search direction obtained in the previous phase. This vector may not lead to a feasible path flow vector since any one of the constraints

\[
x_{k+1}^w = r_w - \sum_{P \in P_w} \hat{p}_P \geq 0, \quad \forall w \in W
\]

(70)
may be violated (particularly when far from the solution). In this case the stepsize should be adjusted so that these constraints are satisfied. This can be done by considering the vector

\[ y_k(\alpha) = [y_k + \alpha d_k]^+ \quad \alpha \geq 0 \]  

(71)

and find the largest stepsize \( \alpha_k \) for which all the constraints

\[ \sum_{p \in p_w} y_k^p(\alpha) \leq r_w \quad \forall w \in W \]  

(72)

are satisfied. The simplest way to determine \( \alpha_k \) is to compute for each OD pair \( w \) the largest stepsize \( \alpha^w_k \) for which (72) is satisfied and obtain \( \alpha_k \) by means of the equation

\[ \alpha_k = \min \{ \alpha^w_k \mid w \in W \} \]  

(73)

One may then successively reduce the value of \( \alpha_k \) by multiplication by a factor less than unity until a sufficient reduction of the objective function is effected in the spirit of the Armijo rule (see [28]).

The stepsize selection method described above can be rigorously shown to lead to convergence of the resulting algorithm by means of a proof which is very similar to one given in [28]. On the other hand for specific applications other stepsize selection methods may be more attractive even though their theoretical properties may not be as reassuring. For example in routing problems arising in communication networks which are solved in real time it is difficult to compute the value of the objective function thereby making line search somewhat impractical.
It is also cumbersome to coordinate the computation of $\alpha_k$ via (73) between all OD pairs. In such cases it is easier to implement other iterations such as for example

$$y_{k+1}^P = [y_k^P + \alpha_k \Delta y_k^P]^*, \quad \forall p \in P, p \neq p_w, w \in W$$

(74)

and forego any tests of reduction of the function value. Our computational experience suggests that iteration (74) is for many networks just as computationally efficient as any other iteration based on line search. On the other hand we have found examples where iteration (74) does not converge to an optimal solution, and therefore cannot recommend it for general networks. The subject of stepsize selection without line search is currently under investigation.

There are a number of convergence and rate of convergence results that one can show for the algorithm described above and its variations. The nature of these results and their proofs are very similar to those given in [28], as well as in other sources [31], [36], and we will not give a complete account. We only mention that it is possible to show that if the stepsize $\alpha_k$ of (73) is used, and if the algorithm is started sufficiently close to a global minimum and a sufficiently accurate solution of the Newton system (58) is obtained via the conjugate gradient method (i.e., the scalar $\beta_k$ in (68) is sufficiently small) then the method converges to a global minimum, and for all $k$ the stepsize $\alpha_k$ will be unity. If in addition $\beta_k + 0$ then the rate of convergence will be superlinear.

We finally mention that in some cases the number of paths in $P_w$ may be very large and it may be unwieldy to keep track of all the path flows.
$x^P$, as for example when $P_w$ is the set of all directed paths joining OD pair $w$. In this case typically the vast majority of path flows at the optimum is zero and it is better to work with a small subset of paths of each OD pair $w$ that carry positive flow. This subset is possible augmented at each iteration by a path of minimum first derivative length which is determined via a shortest path computation (see [13], [15], [16]).
References


