August 1981 LIDS-P-1140

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

$$\text{minimize} \quad J(x) \qquad (1)$$
$$\text{subject to} \quad Ax \leq b,$$

where $J: R^n \to R$, $A : m\text{x}n$, $b \varepsilon R^m$, 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

$$x_{k+1} = x_k + \alpha_k(\bar{x}_k - x_k) \qquad (2)$$

where $\bar{x}_k$ solves

$$\text{minimize} \; \nabla J(x_k)'(x-x_k) + \frac{1}{2}(x-x_k)'H_k(x-x_k) \qquad (3)$$
$$\text{subject to} \quad Ax \leq b,$$

H_k is a positive definite matrix, and α_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 smalll 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

$$\begin{aligned} &\text{minimize} \quad J(x) \\ &\text{subject to} \quad x \geq 0 \end{aligned} \tag{4}$$

it takes the simple form

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

where α_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 non-linear 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 - a fact also observed earlier in a different context by Dembo [29]. A key fact is that the product of the Hessian matrix of the objective function with an arbitrary vector can be obtained by means of graph operations that require relatively little memory storage and computational overhead. 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. Computational results substantiating this fact may be found in [37].

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 R^n is denoted by $|\cdot|$, i.e. for $x = (x^1,\ldots,x^n)$ we write $|x| = [\sum_{i=1}^{n} (x^i)^2]^{1/2}$. The gradient and Hessian of a function $f: R^n \to R$ are denoted by ∇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

$$\begin{aligned} &\text{minimize} \quad J(x) \\ &\text{subject to} \quad x \geq 0, \quad \sum_{i=1}^{n} x^i = r \end{aligned} \tag{6}$$

where $J: R^n \to R$ 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 kth 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_k^n satisfies

$$x_k^n = \max\{x_k^i \mid i = 1,\ldots n\}. \tag{7}$$

Consider a <u>reduced coordinate system</u> in the vector $y \varepsilon R^{n-1}$ given by

$$y = (y^1,\ldots,y^{n-1}) = (x^1,x^2,\ldots,x^{n-1}), \tag{8}$$

denote $y_k = (x_k^1,\ldots,x_k^{n-1})$, and consider the reduced objective function

$$h_k(y) = J(y^1,\ldots,y^{n-1}, r - \sum_{i=1}^{n-1} y^i). \tag{9}$$

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

$$\begin{aligned}&\text{minimize } h_k(y)\\ &y \geq 0\end{aligned} \tag{10}$$

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)x(n-1) positive definite symmetric matrix D_k to be further specified later denote

$$y_k(\alpha) = [y_k - \alpha D_k \nabla h_k(y_k)]^+, \qquad \forall\, \alpha \geq 0 \tag{11}$$

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

$$y_{k+1} = y_k(\alpha_k) \tag{12}$$

where the stepsize α_k is chosen by means of a rule to be specified further later from the range

$$\alpha_k \varepsilon [0, \bar{\alpha}_k] \tag{13}$$

with $\bar{\alpha}_k$ given by

$$\bar{\alpha}_k = \sup \{\alpha \mid \sum_{i=1}^{n-1} y_k^i(\alpha) \leq r\}. \tag{14}$$

[Note that in view of (7), (8), (11), we have $\bar{\alpha}_k > 0$ or $\bar{\alpha}_k = \infty$]. The next vector x_{k+1} generated by the algorithm has coordinates given by

$$x_{k+1}^i = y_{k+1}^i \quad , \quad i = 1,\dots,n \tag{15a}$$

$$x_{k+1}^n = r - \sum_{i=1}^{n-1} y_{k+1}^i \tag{15b}$$

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. Its proof is obtained easily by using Proposition 1 of [28] and the preceding analysis.

Denote

$$I_k^+(x_k) = \{i \mid y_k^i = 0, \ \frac{\partial h_k(y_k)}{\partial y^i} > 0\} \tag{16}$$

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

$$x_k^i(\alpha) = y_k^i(\alpha), \quad i = 1,\dots,n-1 \tag{17}$$

$$x_k^n(\alpha) = r - \sum_{i=1}^{n-1} y_k^i(\alpha). \tag{18}$$

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_k^{ij} of D_k satisfy

$$D_k^{ij} = 0$$

for all $i \varepsilon I_k^+(x_k)$ and $j = 1,\dots,n$ with $i \neq j$.

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

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

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

$$J[x_k(\alpha)] < J(x_k), \qquad \forall \alpha \varepsilon (0,\bar{\alpha}]. \tag{19}$$

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 α_k and associated questions of convergence and rate of convergence which are addressed in [28] for the case of the related problem (4) 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

$$I_k^+ = \{i \mid 0 \leq y_k^i \leq \varepsilon_k^i, \ \frac{\partial h_k(y_k)}{\partial y^i} > 0\} \tag{20}$$

where

$$\varepsilon_k^i = \min\{\varepsilon, s_k^i\} \tag{21}$$

ε is a fixed positive scalar, s_k^i is given by

$$s_k^i = \left| y_k^i - \left[y_k^i - \mu_k^i \frac{\partial h_k(y_k)}{\partial y^i} \right]^+ \right| \tag{22}$$

and μ_k^i are scalar sequences such that

$$\mu_k^i \geq \mu^i > 0, \qquad k = 0,1,\ldots$$

with μ^i being some positive scalars which are fixed throughout the algorithm. This is an antizigzagging 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.

Regarding the choice of the stepsize α_k, there are at least two practical methods that lead to algorithms which are demonstrably convergent. In the first method α_k is chosen according to

$$\alpha_k = \min \{\bar{\alpha}, \bar{\alpha}_k\} \tag{23}$$

where $\bar{\alpha}$ is a fixed positive constant and $\bar{\alpha}_k$ is given by (14). (If D_k is chosen on the basis of second derivatives of the objective function as in the algorithm of the next section the scalar $\bar{\alpha}$ should equal unity). 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 sequence generated by the algorithm are global minima of problem (6). A proof of this fact is obtained by slight modification of the proof of Proposition 2 of [28]. 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 [in view of (11) and (20)] that <u>the portion of the matrix</u> D_k <u>which must be "diagonalized" plays no role near the end of the algorithm</u>. 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 kth 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 ∇h_k. Then the vector y is partitioned as in

$$y = \begin{bmatrix} \tilde{y} \\ \bar{y} \end{bmatrix} \tag{24}$$

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

$$\tilde{H}_k \tilde{d} = -\tilde{g}_k \tag{25}$$

$$\bar{H}_k \bar{d} = -\bar{g}_k \tag{26}$$

where $\tilde{g}_k$ (or $\bar{g}_k$) is the vector with coordinates $\frac{\partial h_k(y_k)}{\partial y^i}$ with $i \varepsilon I_k^+$ (respectively $i \notin I_k^+$), $\tilde{H}_k$ is a <u>diagonal</u> 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, $i \notin I_k^+$. The vector y_{k+1} is then obtained by

$$y_{k+1} = [y_k + \alpha_k d_k]^+ \qquad (27)$$

where α_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 α_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" $\bar{d}_k$ 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 typically 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

$$\bar{z}'\bar{g}_k < 0 \qquad (28)$$

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

$$\bar{H}_k z = -\bar{g}_k, \tag{29}$$

or equivalently the unconstrained minimization problem

$$\min_z \bar{g}_k' z + \frac{1}{2} z'\bar{H}_k z \tag{30}$$

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

$$z_0 = 0, \quad z_{m+1} = z_m + \gamma_m p_m, \quad m = 0,1,\ldots \tag{31}$$

where the conjugage direction sequence $\{p_m\}$ is given recursively by

$$p_0 = -S_k r_0, \; p_m = -S_k r_m + \beta_m p_{m-1}, \quad m = 1,2,\ldots, \tag{32}$$

the residual sequence $\{r_m\}$ is defined by

$$r_m = \bar{H}_k z_m + \bar{g}_k, \quad m = 0,1,\ldots \tag{33}$$

and the scalars γ_m and β_k are given by

$$\gamma_m = \frac{r_m' S_k r_m}{p_m' \bar{H}_k p_m}, \quad m = 0,1,\ldots \tag{34}$$

$$\beta_m = \frac{r'_m S_k r_m}{r'_{m-1} S_k r_{m-1}} , \quad m = 1,2,\ldots \tag{35}$$

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 popular 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 $\bar{y}^i$, $i \notin I_k^+$ evaluated at y_k. There are however other attractive possibilities depending on problem structure (see [33]).

It is easily verified that if $\bar{g}_k \neq 0$, then we have

$$z'_m \bar{g}_k < 0, \quad \forall\, m = 1,2,\ldots$$

so, regardless of how many conjugate gradient iterations are performed, the final approximate solution $\bar{z}$ 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 $\bar{g}_k \neq 0$ 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

$$\text{minimize } J[x(1),\dots,x(m)] \tag{37}$$

$$\text{subject to } x(j) \geq 0, \quad \sum_{i=1}^{n_j} x^i(j) = r_j, \; j = 1,\dots,m$$

where each $x(j)$, $j = 1,\dots,m$ is a vector in R^{n_j}, the function $J: R^{n_1+\dots+n_m} \to R$ is convex and twice continuously differentiable and r_j, $j=1,\dots,m$ 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 $(n_1+\dots+n_m-m)$ while in the process eliminating the m equality constraints $\sum_{i=1}^{n_j} x^i(j) = r(j)$, $j = 1,\dots,m$, [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,...,N and a set of directed links denoted by L. We denote by (i,ℓ) the link from node i to node ℓ, 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 $w \varepsilon W$, we are given a set of directed paths P_w that begin at the origin node and terminate at the destination node. For each $w \varepsilon W$ 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 $p \varepsilon P_w$ corresponding to an OD pair $w \varepsilon W$, we denote by x^p the flow travelling on p. These flows must satisfy

$$\sum_{p \varepsilon P_w} x^p = r_w, \qquad \forall w \varepsilon W \tag{38}$$

$$x^p \geq 0, \qquad \forall \, p \varepsilon P_w, \; w \varepsilon W. \tag{39}$$

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

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

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

where $\delta_p(i,\ell) = 1$ if the path p contains the link (i,ℓ) 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,ℓ) we are given a convex twice continuously differentiable scalar function $D_{i\ell}(f_{i\ell})$ with strictly positive second derivative for all $f_{i\ell} \geq 0$. The objective function is given by

$$D(f) \triangleq \sum_{(i,\ell)\epsilon L} D_{i\ell}(f_{i\ell}). \tag{42}$$

By using (41) we can write the problem in terms of the path flow variables $\mathbf{x}^p$ as

$$\text{minimize} \quad J(x) \triangleq D(Ex) \tag{43}$$

$$\text{subject to} \sum_{p\epsilon P_w} x^p = r_w, \quad \forall\, w\epsilon W$$

$$x^p \geq 0, \quad \forall\, p\epsilon P_w,\ w\epsilon W.$$

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 <u>product of</u>

the matrix $\bar{H}_k$ with various vectors, which is needed for the computation of the residual r_m in (33) and the stepsize γ_m in (34), can be computed by graph type operations without explicitly computing or storing the matrix $\bar{H}_k$.

We now describe the kth 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 $w\varepsilon W$ let p_w be the path carrying maximal flow, i.e.,

$$x_k^{p_w} = \max \{x_k^p \mid p\varepsilon P_w\}, \quad \forall\, w\varepsilon W \tag{44}$$

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

$$y^p = x^p, \quad \forall\, p\varepsilon P_w \text{ with } p \neq p_w \text{ and } w\varepsilon W, \tag{45}$$

and denote by y_k the vector that corresponds to x_k according to this transformation. Consider the reduced objective function $h_k(y) = J(x)$ [cf. (9)] where x has coordinates given by $x^p = y^p$, $\forall p\varepsilon P_w$ with $p \neq p_w$ and $w\varepsilon W$ and

$$x^{p_w} = r_w - \sum_{\substack{p\varepsilon P_w \\ p\neq p_w}} x^p. \tag{46}$$

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

$$l_p = \sum_{(i,\ell)\varepsilon p} D'_{i\ell} \;, \quad \forall\, p\varepsilon P_w, \quad w\varepsilon W, \tag{47}$$

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

It is easily verified that

$$\frac{\partial J(x_k)}{\partial x^p} = l_p \quad , \quad \forall\, p\varepsilon P_w,\ w\varepsilon W \tag{48}$$

and that the gradient of the reduced objective function is given by

$$\frac{\partial h_k(y_k)}{\partial y^p} = l_p - l_{p_w} \quad , \quad \forall\, p\varepsilon P_w,\ w\varepsilon W \tag{49}$$

By differentiating this expression with respect to y^p we also find after a straightforward calculation the diagonal elements of the Hessian $\nabla^2 h_k$

$$\frac{\partial^2 h_k(y_k)}{(\partial y^p)^2} = \sum_{(i,\ell)\varepsilon L_p} D''_{i\ell} \quad , \forall\, p\varepsilon P_w,\ p \neq p_w,\ w\varepsilon W \tag{50}$$

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 $D''_{i\ell}(f_{i\ell}) > 0$ for all $f_{i\ell} \geq 0$ we see that

$$\mu_k^p = \left[\frac{\partial^2 h_k(y_k)}{(\partial y^p)^2} \right]^{-1} > 0,\ \forall p\varepsilon P_w,\ p \neq p_w,\ w\varepsilon W \tag{51}$$

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)]

$$I_k^+ = \{p \mid 0 \leq y_k^p \leq \varepsilon_k^p,\ l_p > l_{p_w},\ p\varepsilon P_w,\ p \neq p_w,\ w\varepsilon W\} \tag{52}$$

where

$$\varepsilon_k^p = \min \{\varepsilon,\ s_k^p\} \tag{53}$$

and

$$s_k^p = |y_k^p - [y_k^p - \mu_k^p(l_p - l_{p_w})]^+| \quad , \quad \forall\, p\varepsilon P_w,\ p \neq p_w,\ w\varepsilon W \tag{54}$$

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 ε and $\mu_k^p(\ell_p-\ell_{p_w})$. 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)]

$$y = \begin{bmatrix} \tilde{y} \\ \bar{y} \end{bmatrix} \tag{55}$$

where $\tilde{y}$ is the vector of path flows y^p with $p\varepsilon I_k^+$ and $\bar{y}$ is the vector of path flows y^p with $p\notin I_k^+$. The search direction d_k, partitioned consistently with (55)

$$d_k = \begin{bmatrix} \tilde{d}_k \\ \bar{d}_k \end{bmatrix} \tag{56}$$

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

$$\tilde{d}_k^p = -\mu_k^p(l_p - l_{p_w}), \quad \forall\, p\varepsilon I_k^+ \tag{57}$$

i.e. the matrix $\tilde{H}_k$ of (25) is set to the diagonal matrix with elements $\dfrac{\partial^2 h_k(y_k)}{(\partial y^p)^2}$ along the diagonal.

For paths $p\notin I_k^+$ the search direction is defined by

$$\bar{H}_k\, \bar{d}_k = -\bar{g}_k \tag{58}$$

where $\bar{g}_k$ is the gradient of h_k with respect to $\bar{y}$ having coordinates $(1_p - 1_{p_w})$, [cf. (49)] and $\bar{H}_k$ is the Hessian matrix of h_k with respect to $\bar{y}$. This equation will be solved (perhaps approximately) by means of the conjugate gradient method described in the previous section [cf. equations (31)-(35)]. As scaling matrix S_k in (32) and (35) we will choose the diagonal matrix with diagonal elements the scalars μ_k^p, $p \notin I_k^+$, $p \neq p_w$, $w \in W$, 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

$$v = \bar{H}_k \Delta y \tag{59}$$

where Δy is any vector of dimension equal to the number of paths p with $p \notin I_k^+$ and $p \neq p_w$, $w \in W$.

A key fact is that <u>in order to compute, for a given Δy, the vector $v = \overline{H}_k \Delta y$ of (59) we need not form explicitly the matrix $\overline{H}_k$ and multiply it with Δy</u>. Indeed consider the following function

$$G_k(\Delta f) = \frac{1}{2} \sum_{(i,\ell)\varepsilon L} (\Delta f_{i\ell})^2 D''_{i\ell} \tag{60}$$

of the incremental flow vector Δf and the corresponding function of the reduced incremental path flow vector Δy

$$M_k(\Delta y) = G_k(E\Delta x) \tag{61}$$

obtained via the transformation

$$\Delta f = E\, \Delta x \tag{62}$$

[cf. (41)] and the transformation

$$\Delta y^p = \Delta x^p, \qquad \forall p\varepsilon P_w,\ p\notin I_k^+,\ p \neq p_w,\ \ w\varepsilon W, \tag{63}$$

$$\Delta x^p = 0, \qquad \forall p\varepsilon I_k^+ \tag{64}$$

$$\Delta x^{p_w} = - \sum_{\substack{p\varepsilon P_w \\ p\neq p_w}} y^p, \qquad \forall w\varepsilon W. \tag{65}$$

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

$$|r_m| \leq \beta_k |r_o| \tag{68}$$

where β_k is some scalar factor less than unity which may depend on the iteration index k.

c) Terminate either as in a) [or as in b)] or if some coordinate of the vector $(\bar{y}+z_m)$ has a negative coordinate, whichever comes first.

Taking $\beta_k = 0$ in (68) means solving the system $H_k \Delta y_k = -\bar{g}_k$ 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 rule for choosing the stepsize α_k. (A result of this type is shown for the unconstrained Newton's method in [36]).

Phase 3: (Determination of the stepsize α_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

$$\hat{y}_{k+1} = [y_k + d_k]^+ \tag{69}$$

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

$$\hat{x}^{p_w}_{k+1} = r_w - \sum_{\substack{p \in P_w \\ p \neq p_w}} \hat{y}^p_{k+1} \geq 0, \qquad \forall\, w \in W \tag{70}$$

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 $\overline{y}$ is equal to the matrix $\overline{H}_k$. For any vector Δy the vector $v = \overline{H}_k \Delta y$ is therefore equal to the gradient $\nabla M_k(\Delta y)$, i.e.

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

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 Δy according to (62)-(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)].

$$v^p = \sum_{(i,\ell)\varepsilon p} D''_{i\ell} f_{i\ell} - \sum_{(i,\ell)\varepsilon p_w} D''_{i\ell} f_{i\ell}$$

$$\forall\, p\varepsilon P_w,\ p\not\varepsilon I_k^+,\ p \neq p_w,\ w\varepsilon W. \tag{67}$$

Thus the products $\overline{H}_k z_m$ and $\overline{H}_k p_m$ 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 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 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

$$\hat{y}_k(\alpha) = [y_k + \alpha d_k]^+, \qquad \alpha \geq 0 \tag{71}$$

and finding the largest stepsize $\bar{\alpha}_k$ for which all the constraints

$$\sum_{\substack{p\varepsilon P_w \\ p\neq p_w}} y_k^p(\alpha) \leq r_w, \qquad w\varepsilon W \tag{72}$$

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

$$\bar{\alpha}_k = \min \{\bar{\alpha}_k^w \mid w\varepsilon W\}. \tag{73}$$

One may then successively reduce the value of $\bar{\alpha}_k$ 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]).

There are a number of convergence and rate of convergence results that one can show for the algorithm described above and its variations. These results are similar in nature to corresponding results given in [28] and 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 $\bar{\alpha}_k$ 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 β_k in (68) is sufficiently small] then the method converges to a global minimum, and for all k the stepsize $\bar{\alpha}_k$ will be unity. If in addition $\beta_k \to 0$ 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 unwieldly 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 augmented at each iteration by a path of minimum first derivative length (see [13], [15], [16]).

References

[1] A. A. Goldstein, "Convex Programming in Hilbert Space", Bull. Amer. Math. Soc., Vol. 70, 1964, pp. 709-710.

[2] E. S. Levitin and B. T. Poljak, "Constrained Minimization Problems", U.S.S.R. Comput. Math. and Math. Phys., Vol. 6, 1976, pp. 1-150.

[3] J. C. Dunn, "Newton's Method and the Goldstein Step-Length Rule for Constrained Minimization Problems", SIAM J. Control and Optimization, Vol. 6, 1980, pp. 659-674.

[4] U. M. Garcia-Palomares and O. L. Mangasarian, "Superlinearly Convergent Quasi-Newton Algorithms for Nonlinearly Constrained Optimization Problems", Math. Programming, Vol. 11, 1976, pp. 1-13.

[5] M. Frank and P. Wolfe, "An Algorithm for Quadratic Programming", Nav. Res. Logistics Quart., Vol. 3, 1956, pp. 149-154.

[6] L. Fratta, M. Gerla, and L. Kleinrock, "The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design", Networks, Vol. 3, 1973, pp. 97-133.

[7] S. C. Dafermos, "An Extended Traffic Assignment Model with Applications to Two-Way Traffic", Transportation Science, Vol. 5, 1971, pp. 366-389.

[8] M. Florian, "An Improved Linear Approximation Algorithm for the Network Equilibrium (Packet Switching) Problem", Proc. of 1977 IEEE Conf. on Dec. and Control, New Orleans, La., 1977, pp. 812-818.

[9] M. D. Cannon and C. D. Cullum, "A Tight Upper Bound on the Rate of Convergence of the Frank-Wolfe Algorithm", SIAM J. Control and Optim., Vol. 6, 1968, pp. 509-516.

[10] J. C. Dunn, "Rates of Convergence of Conditional Gradient Algorithms Near Singular and Nonsingular Extremals", SIAM J. Control and Optim., Vol. 17, 1979, pp. 187-211.

[11] R. G. Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation", IEEE Trans. on Communications, Vol. COM-25, 1978, pp. 73-85.

[12] D. P. Bertsekas, "Algorithms for Nonlinear Multicommodity Network Flow Problems", International Symposium on Systems Optimization and Analysis, A. Bensoussan and J. L. Lions (eds.), Springer-Verlag, 1979, pp. 210-224.

[13] D. P. Bertsekas, "A Class of Optimal Routing Algorithms for Communication Networks", Proc. of 5th International Conference on Computer Communication (ICCC-80), Atlanta, Ga., Oct. 1980, pp. 71-76.

[14] E. Gafni, "Convergence of a Routing Algorithm", Lab. for Information and Decision Systems Report R-907, M.I.T., Cambridge, Mass., May 1979.

[15] D. P. Bertsekas and E. Gafni, "Projection Methods for Variational Inequalities with Application to the Traffic Assignment Problem", Math. Prog. Study, 17, D. C. Sorensen and R. J.-B. Wets (eds.), North-Holland, Amsterdam, 1982, pp. 139-159.

[16] H. Z. Aashtiani, "The Multi-Model Traffic Assignment Problem", Ph.D. Thesis, Sloan School of Management, M.I.T., Cambridge, Mass., May, 1979.

[17] S. C. Dafermos, "Traffic Equilibrium and Variational Inequalities", Transp. Science, Vol. 14, 1980, pp. 42-54.

[18] D. P. Bertsekas, "On the Goldstein-Levitin-Poljak Gradient Projection Method", IEEE Trans. Aut. Control, Vol. AC-21, 1976, pp. 174-184.

[19] J. C. Dunn, "Global and Asymptotic Convergence Rate Estimates for a Class of Projected Gradient Processes", SIAM J. Control and Opt., Vol. 19, 1981, pp. 368-400.

[20] J. B. Rosen, "The Gradient Projection Method for Nonlinear Programming, Part I: Linear Constraints", J. Soc. Ind. Appl. Math., Vol. 8, 1960, pp. 181-217.

[21] W. I. Zangwill, Nonlinear Programming, Prentice Hall, Englewood Cliffs, N.J., 1969.

[22] P. E. Gill and M. Murray, eds., Numerical Methods for Constrained Optimization, Academic Press, New York, Ch. 2,3, 1974.

[23] D. Goldfarb, "Extension of Davidon's Variable Metric Algorithm to Maximization Under Linear Inequality and Equality Constraints", SIAM J. Applied Math., 17, (1969), pp. 739-764.

[24] M. L. Lenard, "A Computational Study of Active Set Strategies in Nonlinear Programming with Linear Constraints", Math. Prog., 16, 1979, pp. 81-97.

[25] D. G. Luenberger, Introduction to Linear Nonlinear Programming, Addison-Wesley, Reading, 1973, Ch. 11.

[26] M. Florian and S. Nguyen, "A Method for Computing Network Equilibrium with Elastic Demand", Trans. Sci., Vol. 8, 1974, pp. 321-332.

[27] R. S. Dembo and J. G. Klincewicz, "A Scaled Reduced Gradient Algorithm for Network Flow Problems with Convex Separable Costs", Math. Programming Studies (to appear).

[28] D. P. Bertsekas, "Projected Newton Methods for Optimization Problems with Simple Constraints", SIAM J. Control and Optimization, Vol. 20, 1982, pp. 221-246.

[29] R. S. Dembo, Seminar at M.I.T., Nov. 1979, and private communication.

[30] E. Polak, Computational Methods in Optimization: A Unified Approach, Academic Press, New York, 1971.

[31] J. M. Ortega and W. C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, N.Y., 1970.

[32] M. R. Hestenes, Conjugate Direction Methods in Optimization, Springer-Verlag, N.Y., 1980.

[33] D. P. Bertsekas, "Partial Conjugate Gradient Methods for a Class of Optimal Control Problems", IEEE Trans. on Aut. Control, Vol. AC-19, 1974, pp. 209-217.

[34] R. G. Gallager and S. J. Golestaani, "Flow Control and Routing Algorithms for Data Networks", Proc. of the Fifth International Conference on Computer Communication (ICCC-80), Atlanta, Ga., Oct. 1980, pp. 779-784.

[35] H. Z. Aashtiani and T. L. Magnanti, "Equilibria on a Conjested Transportation Network", SIAM J. of Algebraic and Discrete Math., Vol. 2, 1981, pp. 213-226.

[36] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, "Inexact Newton Methods" Yale School of Organization and Management Working Paper #47, April 1980.

[37] E. M. Gafni, "The Integration of Routing and Flow Control for Voice and Data in a Computer Communication Network", Ph.D. Dissertation, Dept. of Electr. Eng. and Computer Science, M.I.T., Cambridge, Mass., Aug. 1982.