**Title:** A Class of Optimal Routing Algorithms for Communication Networks

**Author(s):** Dimitri P. Bertsekas

**Performing Organization Name and Address:**
Massachusetts Institute of Technology
Laboratory for Information and Decision Systems
Cambridge, MA 02139

**Controlling Office Name and Address:**
Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

**Distribution Statement (of the Report):**
Approved for public release; distribution unlimited.

**Distribution Statement (of the abstract entered in Block 20, if different from Report):**

**Supplementary Notes:**
ICCC 80 Atlanta, Ga., Oct. 1980

**Abstract:**
We describe an algorithm for minimum delay routing in a communication network. During the algorithm each node maintains a list of paths along which it sends traffic to each destination together with a list of the fractions of total traffic that are sent along these paths. At each iteration a minimum marginal delay path to each destination is computed and added to the current list if not already there. Simultaneously the corresponding fractions are updated in a way that reduces average delay per message. The algorithm is...
capable of employing second derivatives of link delay functions thereby providing automatic scaling with respect to traffic input level. It can be implemented in both a distributed and a centralized manner, and it can be shown to converge to an optimal routing at a linear rate.
A CLASS OF OPTIMAL ROUTING ALGORITHMS
FOR COMMUNICATION NETWORKS.

by

Dimitri P. Bertsekas

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, Mass. 02139

† Work supported by Grants ONR N00014-75-C-1183 and NSF ENG-7906332.

FARPA under 3145

4/10/50
ABSTRACT

We describe an algorithm for minimum delay routing in a communication network. During the algorithm each node maintains a list of paths along which it sends traffic to each destination together with a list of the fractions of total traffic that are sent along these paths. At each iteration a minimum marginal delay path to each destination is computed and added to the current list if not already there. Simultaneously the corresponding fractions are updated in a way that reduces average delay per message. The algorithm is capable of employing second derivatives of link delay functions thereby providing automatic scaling with respect to traffic input level. It can be implemented in both a distributed and a centralized manner, and it can be shown to converge to an optimal routing at a linear rate.
I. INTRODUCTION

We consider the problem of optimal quasistatic routing of data in a communication network with slowly varying input traffic statistics. The type of physical situation underlying the problem is the one described in Gallager [1] and we refer to that paper for further discussion. We describe an algorithm for solving the corresponding optimization problem in either a distributed or a centralized manner. The algorithm is similar to Gallager's method [1] and its generalized versions [2] in that it relates to the gradient projection method for nonlinear programming (Goldstein [3], Levitin-Poljak [4] - see Bertsekas [2] for a related discussion). The algorithm given here is different primarily in that it operates in the space of path flows rather than in the space of link flows. Furthermore it utilizes a shortest path computation to obtain a search direction rather than an upstream summation of link marginal delays. As a result the amount of computation per iteration is typically smaller for our algorithm. Our algorithm is also well suited for virtual circuit networks since it operates directly in terms of the quantities of interest - the path flows. In this respect it is similar to a routing algorithm by Segall [5]. The algorithm bears also a close relationship with an algorithm developed in a different context by Aashtiani [6] which utilizes second derivatives. Aashtiani's algorithm need not converge to an optimal routing but, for problems where it does converge, the corresponding computational results have been very encouraging.

Our algorithm is also similar to the flow deviation method (Fratta, Gerla, and Kleinrock [7]) and the extremal flow method (Cantor and Gerla [8]), in that at each iteration it involves the same type of shortest
path computation. However, the first of these methods is characterized by sublinear and hence slow convergence rate ([9], [10]), while the second requires the solution of a nonlinear programming problem at each iteration. By contrast, the convergence rate of our method is linear, while the computational requirements per iteration are rather modest. Furthermore, within the context of our method it is possible to employ second derivatives in a manner that is reminiscent of Newton's method. Computational results for related methods that employ second derivatives [11], [12] have been very favorable and this provides grounds for optimism that the same will be true for corresponding versions of our method. An additional benefit of the use of second derivatives is that it provides automatic stepsize scaling with respect to level of traffic input as has been shown computationally within the context of generalized versions of Gallager's method [11]. This advantage is of crucial importance if the algorithm is operated in a distributed manner.

We consider a network consisting of $N$ nodes denoted by $1, 2, \ldots, N$ 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 consider the following multicommodity flow problem in the vector of total link flows $f = \{ f_{ik} | (i, k) \in L, \ j = 1, \ldots, N \}$:

\begin{align}
\text{(1)} \quad \text{minimize} & \quad D(f) \\
\text{subject to} & \quad \sum_{k \in O(i)} f_{ik}(j) - \sum_{m \in I(i)} f_{mi}(j) = r_i(j), \ \forall i, j = 1, \ldots, N \ i \neq j
\end{align}

$$f_{ik}(j) \geq 0, \ \forall (i, k) \in L, \ i, j = 1, \ldots, N$$

$$f_{jl}(j) = 0 \ \forall (j, l) \in L, \ j = 1, \ldots, N$$

Here $f_{ik}(j)$ is the flow on link $(i, k)$ destined for node $j$, $f_{il} = \sum_{j=1}^{N} f_{ik}(j)$ is the total flow on link $(i, l)$, $O(i)$ and $I(i)$ are the sets of nodes $k$ for which
(i,ℓ)∈L and (ℓ,i)∈L respectively, and, for i ≠ j, \( r_i(j) \) is a given traffic input at node \( i \) destined for \( j \). Each link \((i,ℓ)\) has associated with it a number \( C_{iℓ} \), referred to as the capacity of the link which is assumed positive or \( +\infty \). The standing assumptions throughout the paper are:

a) \( r_i(j) > 0, \quad \forall i,j = 1,...,N, \quad i ≠ j \).

b) The real valued function \( D \) is defined on the set

\[
S = \{f|0 ≤ f_{iℓ} < C_{iℓ}, \quad \forall (i,ℓ)∈L\}
\]

and is convex and twice continuously differentiable. Furthermore \( \lim_{f→+\infty} D(f) = +\infty \),

\[
\frac{∂²D(f)}{∂f_{iℓ}²} > 0,
\]

and the Hessian matrix of \( D \) is positive definite for all \( f∈S \) and \((i,ℓ)∈L\).

Problem (1) is formulated in the space of link flows. We consider an equivalent formulation in the space of path flows. Consider the set of all active origin-destination (OD) pairs

\[
A = \{(i,j)|r_i(j) > 0, \quad i,j = 1,...,N\}.
\]

For each \( a = (i,j)∈A \) we also write \( r_i(j) = r_a \). For each OD pair \( a = (i,j)∈A \) let \( P_a \) be the set of directed paths with no repeated nodes originating at \( i \) and terminating at \( j \). For \( a = (i,j)∈A \) and a path \( p∈P_a \) we denote by \( h_p \) the flow originating at \( i \) destined for \( j \) and travelling along the path \( p \). Let \( h = \{h_p | p∈P_a, \quad a∈A\} \) be the vector of path flows and \( H \) be the set of all feasible path flow vectors

\[
H = \{h|h_p > 0, \quad \sum_{p∈P_a} h_p = r_a, \quad \forall p∈P_a, \quad a∈A\}.
\]

For each \( h∈H \) there is a corresponding flow vector \( f \) satisfying the constraints
of problem (1) and related to $h$ linearly by means of the equation

$$f_{il} = \sum_{a \in A} \sum_{p \in \mathcal{P}_a} \delta_p(i,l)h_p, \quad \forall (i,l) \in L$$

when $\delta_p(i,l) = 1$ if the path $p$ contains the link $(i,l)$ and $\delta_p(i,l) = 0$ otherwise. Let us write (4) as

$$f = Eh,$$

where $E$ is the appropriate matrix, and let us consider the function

$$J(h) = D(Eh),$$

and the problem

$$\text{minimize } J(h)$$
$$\text{subject to } h \in H.$$  

It is easy to show that problem (7) is equivalent to problem (1) in the sense that an optimal solution $h^*$ of problem (7) yields an optimal solution \{$(f^*_l(j))|(i,l) \in L, \ j = 1, \ldots, N$\} of (1) via the relation

$$f^*_l(j) = \sum_{(i,j) \in A} \sum_{p \in \mathcal{P}_{(i,j)}} \delta_p(i,l)^*h^*_p.$$

This is based on the assumption $\frac{\partial D}{\partial f_{il}} > 0$ which precludes the existence of optimal sets of flows that contain loops.

The algorithm of this paper solves numerically problem (7) but its definition, as well as its convergence and rate of convergence properties depend crucially on assumption (b) made earlier regarding the function $D$. In the next section we describe the algorithm for the case where there is a single OD pair. The generalization to the case of several OD pairs and the corresponding convergence and rate of convergence results are given in Section 3. In Section 4 we describe briefly how the algorithm can be operated in a distributed manner.
Proofs of various statements and results of this paper are generally omitted due to space limitations. However, the reader who is familiar with recent work on optimal routing should be able to convince himself of the validity of the algorithm and the corresponding convergence results. References where appropriate are provided as an aid in this respect.

2. The Algorithm for a Single OD Pair

Consider the special case of problem (7) where there is a single OD pair \((i, j) = a\). Problem (7) is written then as

\[
\begin{align*}
\text{minimize} & \quad J(h) = D(Eh) \\
\text{subject to} & \quad \sum_{p \in P_a} h_p = r_a \\
& \quad h_p > 0 \quad \forall p \in P_a 
\end{align*}
\]

The algorithm is initiated with a subset of paths \(P_o \subset P_a\) and a set of path flows \(h^o\) which may be nonzero only on the paths in \(P_o\), i.e.,

\[
h^o_p = 0 \quad \text{if} \quad p \notin P_o
\]

and are such that the corresponding set of total link flows \(f^o = E h^o\) belongs to the set \(S\) of (2), i.e.,

\[
f^o \in S
\]

At the \(k\)th iteration we have a subset of paths \(P^k_a \subset P_a\) and corresponding sets of path flows \(h^k\) and total link flows \(f^k\) satisfying

\[
h^k_p = 0 \quad \text{if} \quad p \notin P^k_a \quad \text{and} \quad f^k \in S
\]

We compute a shortest path \(p^k \in P_a\) by means of a shortest path algorithm using as length of each link \((l, m) \in L\) the partial derivative \(\frac{\partial D(f^k)}{\partial f_{lm}}\).

We set

\[
P^{k+1}_a = P_a \cup \{p^k\}
\]
(Note here that \( k \) may already belong to \( P^k_a \)). We then solve the quadratic programming problem

\[
\text{minimize } \sum_{p \in P^{k+1}_a} \left( d^k_p \left( h^k_p - h^k_p \right) + \frac{1}{2} \mu^k_p \left( h^k_p - h^k_p \right)^2 \right)
\]

subject to \( \sum_{p \in P^{k+1}_a} h^k_p = r_a, \quad h^k_p \geq 0, \quad \forall p \in P^{k+1}_a \),

where for each \( p \in P^{k+1}_a \), \( d^k_p \) is the total length of the path \( p \) with respect to link lengths \( \frac{\partial D(f^k)}{\partial x_m} \), i.e.

\[
\frac{d^k_p}{\partial x_m} = \sum_{(\ell, m) \in L} \delta_p(\ell,m) \frac{\partial D(f^k)}{\partial x_m},
\]

and \( \mu^k_p \) is a positive scalar. If \( h^k_p \) is the solution of problem (13) we set for \( p \in P_a \)

\[
h^{k+1}_p = \begin{cases} h^k_p & \text{if } p \in P^{k+1}_a \\ 0 & \text{if } p \notin P^{k+1}_a \end{cases}
\]

In the way of explanation we mention that there holds

\[
d^k_p = \frac{\partial J(h^k)}{\partial h^k_p}, \quad \forall p \in P^{k+1}_a
\]

[cf. (4), (5), (6) and (14)]. As a result the subproblem (13) may be viewed as a quadratic approximation of the original problem (8) and the algorithm may be viewed as a version of the Goldstein-Levitin-Poljak gradient projection method (see [2]) - the only different being that problem (13) involves only the paths in \( P^{k+1}_a \) rather than the entire set of paths \( P_a \). Note that problem (13) involves a single equality constraint and can be solved by essentially analytical means (compare with [2]). It is possible to show by using rather standard
nonlinear programming arguments that there exists a scalar $\mu > 0$ such that if
\begin{equation}
(16) \quad \mu \leq \frac{\mu_k}{p}, \quad \forall p \in P^{k+1}_a, \quad k = 0, 1, \ldots
\end{equation}
and the sequences $\{u^k_p\}$ are all bounded, then the generated sequence $\{f^k\}$ satisfies $\{f^k\} \subseteq S$ and converges to the unique optimal solution of problem (1).

A choice of $\mu^k_p$ based on second derivatives is given by
\[ \mu^k_p = \eta \frac{3^2J(h^k_p)}{\partial h^2_p} = \eta \sum_{(m,\ell) \in L} \frac{3^2D(f^k_p)}{\partial f^2_{m\ell}} \delta_p(m,\ell) \]
where $\eta$ is a positive scalar. When $\eta = 1$ the resulting algorithm represents a diagonal approximation to Newton's method. Algorithms of this type with $\eta$ near unity have been shown computationally to be quite successful in a related context [11], [12]. Another possible choice is
\[ \mu^k_p = \sum_{(m,\ell) \in L} \eta^k_{m\ell} \frac{3^2D(f^k_p)}{\partial f^2_{m\ell}} \delta_p(m,\ell) \]
where $\eta^k_{m\ell}$ is the number of paths in $P^{k+1}_a$ that contain link $(m,\ell)$.

With this choice it may be shown that the objective function of problem (13) is an upper bound to the one corresponding to Newton's method applied to problem (8). This in turn can be used to show that if $f^k$ is near the optimum, then the kth iteration leads to guaranteed reduction of the objective function.

The algorithm just described does not involve a search for an appropriate stepsize along the descent direction $(h^k - h^k)$, and as such it is more suitable for distributed operation. When the algorithm is operated in a centralized manner it is possible to incorporate a linear
search of the type that is usual in many nonlinear programming algorithms thereby guaranteeing a reduction of the objective function value at each iteration (compare with [2]).

3. The Algorithm for Several OD Pairs

In the general case where there are several OD pairs the algorithm consists of several simultaneous OD pair iterations of the type described in the previous section. The set of OD pairs \( A \) is partitioned in a union of disjoint subsets \( A_1, A_2, \ldots, A_n \) where \( n \) is some positive integer. The algorithm operates in cycles of \( n \) iterations. In the first iteration of each cycle a single iteration of the algorithm of the previous section is carried out simultaneously for all OD pairs in \( A_1 \). In the second iteration of the cycle the same thing is done for all OD pairs in \( A_2 \), and so on until the OD pairs in \( A_n \) are iterated upon. At this point the \( n \)-iteration cycle is completed and a new cycle begins with the OD pairs in \( A_1 \).

Among the possible partitions of the set \( A \) the two extreme cases are when \( n = 1, A_1 = A \), and when \( A_1 \) consists of a single OD pair. Other possibilities of interest are when \( n = N \) and \( A_1 \) consists of all OD pairs corresponding to the same origin \( i \) or the same destination \( i \).

Any one of these algorithms can be shown to generate sequences \( \{f^k\} \) that converge to the unique optimal solution of problem (1) under the assumption that the sequences \( \{\mu_p^k\} \) are bounded and satisfy (16) for a sufficiently large lower bound \( \mu \).

The proof of this fact is similar but actually simpler than Gafni's convergence proof [13] for a related algorithm that operates in the space.
of link flows. Regarding rate of convergence it is possible to show for 
the case where \( n = 1 \), \( A_1 = A \) that \( \{f^k\} \) converges to the optimal solution 
at a linear rate. A proof of this fact for the case where \( \mu_p^k \) is a 
constant may be found in Bertsekas and Gafni [14]. It is our conjecture 
that the convergence rate is linear even when \( n > 1 \) but this remains 
to be shown.

4. Operating the Algorithm in a Distributed Manner

It is possible to distribute the computation involved in each iteration 
of the algorithm among the nodes of the network. Each node \( i \) measures 
\( r_{ij} \) for all destinations \( j \) and receives periodically the values of the 
total flows \( f_{xm} \) of all links \((\ell,m)\). Node \( i \) can then execute the 
portion of the iteration of the algorithm that corresponds to OD pairs 
for which \( i \) is the origin. This involves computation of shortest 
paths from \( i \) to all destinations and an adjustment of the flows of 
the currently active paths according to (13), (15). Each node \( i \) is 
also involved in monitoring the average link flows of outgoing links \( f_{i\ell} \). 
The values of these flows are broadcast to all other nodes at the 
beginning of each iteration by either a flooding scheme, or by means of 
a spanning tree.

A distributed algorithm of the type just described resembles the 
current ARPANET routing algorithm [15] in that information providing a 
length for each link is broadcast throughout the network, each node 
computes shortest paths from itself to each destination on the basis of 
these lengths, and shifts flow to the shortest paths. In the ARPANET 
algorithm all flow is shifted to the shortest path at each iteration. 
In our algorithm a portion of the flow is retained in previous shortest
paths and, if this is done properly, the resulting flow pattern converges asymptotically to the optimal. By contrast an algorithm of the ARPANET type cannot provide optimal routing since it precludes the possibility of sending data along more than one paths for any single OD pair.

A certain amount of synchronization of link flow broadcasts is necessary in our algorithm. It would be interesting to know what would happen if these broadcasts and the corresponding shortest path computations and flow adjustments were carried out asynchronously as in the ARPANET algorithm. Such an algorithm would certainly offer important practical advantages but the associated questions of convergence are quite difficult and as yet unexplored.
References


Distribution List

Defense Documentation Center
Cameron Station
Alexandria, Virginia 22314
12 Copies

Assistant Chief for Technology
Office of Naval Research, Code 200
Arlington, Virginia 22217
1 Copy

Office of Naval Research
Information Systems Program
Code 437
Arlington, Virginia 22217
2 Copies

Office of Naval Research
Branch Office, Boston
495 Summer Street
Boston, Massachusetts 02210
1 Copy

Office of Naval Research
Branch Office, Chicago
536 South Clark Street
Chicago, Illinois 60605
1 Copy

Office of Naval Research
Branch Office, Pasadena
1030 East Greet Street
Pasadena, California 91106
1 Copy

Naval Research Laboratory
Technical Information Division, Code 2627
Washington, D.C. 20375
6 Copies

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps (Code RD-1)
Washington, D.C. 20380
1 Copy
Office of Naval Research
Code 455
Arlington, Virginia 22217

Office of Naval Research
Code 458
Arlington, Virginia 22217

Naval Electronics Laboratory Center
Advanced Software Technology Division
Code 5200
San Diego, California 92152

Mr. E. H. Gleissner
Naval Ship Research & Development Center
Computation and Mathematics Department
Bethesda, Maryland 20084

Captain Grace M. Hopper
NAICOM/MIS Planning Branch (OP-916D)
Office of Chief of Naval Operations
Washington, D.C. 20350

Advanced Research Projects Agency
Information Processing Techniques
1400 Wilson Boulevard
Arlington, Virginia 22209

Dr. Stuart L. Brodsky
Office of Naval Research
Code 432
Arlington, Virginia 22217

Captain Richard L. Martin, USN
Commanding Officer
USS Francis Marion (LPA-249)
FPO New York 09501