Optimal Dynamic Routing in Single Commodity Networks by Iterative Methods

GALEN SASAKI AND BRUCE HAJEK, SENIOR MEMBER, IEEE

Abstract—**In this paper, we present iterative methods for finding optimal state-dependent routing strategies in single commodity networks. The key to our method is to show that there exists a family of optimization problems with convex cost and linear constraints that have solutions that can be converted into an optimal routing strategy by way of a *flow relaxation* transformation. These problems, when solved by certain iterative algorithms, lead to different convergence rates. In particular, one of the problems has quadratic cost.**

To solve one of these optimization problems, we use an iterative projected descent direction algorithm due to Bertsekas. We present an alternative to the Armijo-like step size rule of the algorithm, which we believe is more robust. Also included are Newton-like descent directions that take a reasonable amount of time to compute. Finally, some results of our computer experiments are summarized.

I. Introduction

A. Purpose

In this paper, we present dynamic routing algorithms for single commodity communication networks. The model used for the network is a special case of the model introduced by Segall [1] and is the same one used in [2] and [3]. Feit [4] also considered the network model, but for multiple commodities. This problem is dynamic in the sense that the control depends on the state of the network.

The key to our algorithm is that there exists a family of *initial flow* optimization problems with convex cost and linear constraints whose members have solutions that can be converted to an optimal routing strategy through a simple *flow relaxation* transformation. We show that some of the initial flow problems can be solved using Bersekas' projected descent direction algorithm for problems with simple constraints [5, subsect. 1.5]. Since this algorithm is an iterative descent algorithm, it may not find an optimal solution in finite time, although it gets arbitrarily close. This property, coupled with the fact that algorithm OPTFLO [2] can solve one of the initial flow problems in $O(|N|^4)$ time (where $|N|$ is the number of nodes in the network), makes the iterative method seem inefficient.

However, in cases where we will be satisfied with a near optimal routing strategy, the iterative descent algorithm may be useful. To see this, first suppose we have an initial flow problem $\bar{P}$ which has the property that the cost of a feasible vector for $\bar{P}$ reflects the performance of the network when the routing strategy is the flow relaxation transformation of the vector. Then with each iteration of the iterative descent algorithm, we get a succeeding feasible vector f with lower cost and which has a flow relaxation that will tend to be a better routing strategy than the flow relaxation of the previous feasible vector. If certain conditions on the cost of $\bar{P}$ hold, then we can expect each iteration of the algorithm to take only $O(|N|^2)$ time. The iterative method may be well suited for updating solutions of $\bar{P}$ when small perturbations in the state of the network occur.

This paper is organized as follows. The problem and the concept of *flow relaxation* are presented in the next two subsections. In Section II, the family of initial flow problems is given, where one such problem has quadratic cost. In Section III, we present Bertsekas' projected descent direction algorithm for optimization problems subject to simple constraints. This algorithm uses an Armijo-like rule to determine step sizes. We modify this rule so that it does not seem to work much worse and at times can work much better than the original rule.

There are conditions on the cost of the problem and on the descent direction that together are sufficient for Bertsekas' algorithm to converge to an optimal solution. Included in Section III are methods to modify the problem and to choose descent directions so that these conditions hold. In Section IV, some results of computer experiments are presented. Finally, in Section V, a conclusion is given.

B. The Network Model

A single destination network N is a quadruple (N, L, C, δ) where (N, L) is a directed graph with a set of nodes N and a set of directed links L, δ is a distinguished node in N called the destination, and $C = (C_e : e \in L)$ is a capacity assignment vector so that $C_e \geq 0$ for each link e. For each node i in N, $x_i(t)$ is a real value which denotes the amount of traffic at node i at time t (measured in bits, packets, vehicles, or messages, for example).

A *demand* for the network is a pair $(x(0), r)$ where, for each i in N, $x_i(0)$ denotes the (nonnegative) initial amount of traffic at node i and r_i denotes the (nonnegative) rate at which traffic flows into node i from outside the network. By convention, $x_\delta(t) = r_\delta = 0$ for all $t \geq 0$. Let B be the $|N| \times |L|$ matrix such that for a vector $f = (f_e : e \in L)$, Bf is the vector with ith coordinate

$$Bf_i = \sum_{k:(k,i)\in L} f_{ki} - \sum_{j:(i,j)\in L} f_{ij} \qquad \text{for } i \in N.$$

Note that if f_e is the constant (nonnegative) rate of flow on link e, then $Bf_i + r_i$ is the net rate of flow going into node i. Given a control $u = (u_e(t) : e \in L, t \geq 0)$ where $u_e(t)$ denotes the instantaneous flow on link e at time t, the state equations of the network are

$$\dot{x}_i(t) = (Bu(t) + r)_i \qquad i \neq \delta.$$

A state trajectory $(x(t) : t \geq 0)$ is well defined by (the integrated version of) the state equation above for any demand $(x(0), r)$ and any control u in the set

$$U = \{u : u \text{ is a measurable function on } \mathbb{R}_+ \text{ and } 0 \leq u \leq C\}.$$

Paper approved by the Editor for Computer Communication Theory of the IEEE Communications Society. Manuscript received December 29, 1985; revised April 29, 1987. This work was supported by the Office of Naval Research under Contract U.S. NAVY N00014-82-K0359. This paper was presented in part at the 1985 Conference on Information Sciences and Systems, Baltimore, MD, March 1985.

G. Sasaki was with the Department of Electrical Engineering and the Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801. He is now with the Department of Electrical and Computer Engineering, University of Texas, Austin, TX.

B. Hajek is with the Department of Electrical Engineering and the Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801.

IEEE Log Number 8717085.

Let $x^u(t)$ be the state trajectory for control u at time t. Then we term a control u in U *admissible* for the demand $(x(0), r)$ if $x^u(t) \geqslant 0$ for all $t \geqslant 0$.

C. Admissible Controls Obtained by Flow Relaxation

Suppose u is a control in U which is not necessarily admissible for a given demand $(x(0), r)$. Then a control $\tilde{u}$ is defined to be a *relaxation* of u for $(x(0), r)$ if $\tilde{u}$ is admissible for $(x(0), r)$ and if, using x to denote the corresponding (nonnegative) state trajectory, the following conditions hold for each $t \geqslant 0$:

$$0 \leqslant \tilde{u}(t) \leqslant u(t)$$

$$\tilde{u}_{ij}(t) = u_{ij}(t) \qquad \text{if } x_i(t) > 0.$$

Thus, a relaxed control differs from the original control only in that nodes with zero traffic may have reduced flow on links directed outwards, which may be necessary to keep the coordinates of x nonnegative.

Given a vector f with $0 \leqslant f \leqslant C$, we let u^f denote the constant control with initial value f—thus $u^f(t) = f$ for all $t \geqslant 0$. A relaxation $\tilde{u}^f$ of u^f can be computed in $O(|N|^2|L|)$ steps, but more importantly, it can be easily computed in a distributed way [2]. Given a network N and demand $(x(0), r)$, it is shown in [2, Corollary 2] that if f is chosen properly, then $\tilde{u}^f$ solves the problem

$$\min \left\{ \sum_{i \in N-\delta} x_i(t) : u \text{ is admissible} \right\}$$

for any $t > 0$. The choice of the initial flow f is made independently of t, and $\tilde{u}^f$ can be an arbitrary relaxation of u^f. Furthermore, an optimal f can be found by solving a certain convex optimization problem [2]. The overall process is illustrated in Fig. 1.

In this paper, we explore iterative methods for finding optimal initial flows. As a first step, we provide in the next section a variety of convex optimization problems, some leading to better rates of convergence, for finding optimal initial flows.

II. Initial Flow Problems Leading to Minimum Traffic in Network at All Times

Suppose $0 \leqslant f \leqslant C$ and the (possibly not admissible) constant control u^f are used. Consider node i in $N - \delta$. The node initially has $x_i(0)$ units of traffic, and traffic forever enters the node at net rate $Bf_i + r_i$, so the node has $x_i(0) + t(Bf_i + r_i)$ units of traffic at time t. If $Bf_i + r_i < 0$, the amount of traffic in the node hits zero at time $-x_i(0)/(Bf_i + r_i)$. When the admissible control $\tilde{u}^f$ is used, then the flow out of node i is reduced, but only after the node empties. So for control $\tilde{u}^f$, using the notation $[\beta]_+ = \max(0, \beta)$, $[x_i(0) + t(Bf_i + r_i)]_+$ upper bounds the amount of traffic in the node at time t, for any $t > 0$.

Suppose g is a function on $(0, \infty)$ such that

g is nonnegative, right- or left-continuous and

$$\int_0^s g(t)\, dt < +\infty \qquad \text{for all } s > 0 \tag{2.1}$$

and define a function d by

$$d(\sigma, \theta) = \int_0^\infty g(t)[\theta + \sigma t]_+ \, dt. \tag{2.2}$$

Then for any node i, $d(Bf_i + r_i, x_i(0))$ is an upper bound on

$$\int_0^\infty g(t) x_i(t)\, dt$$

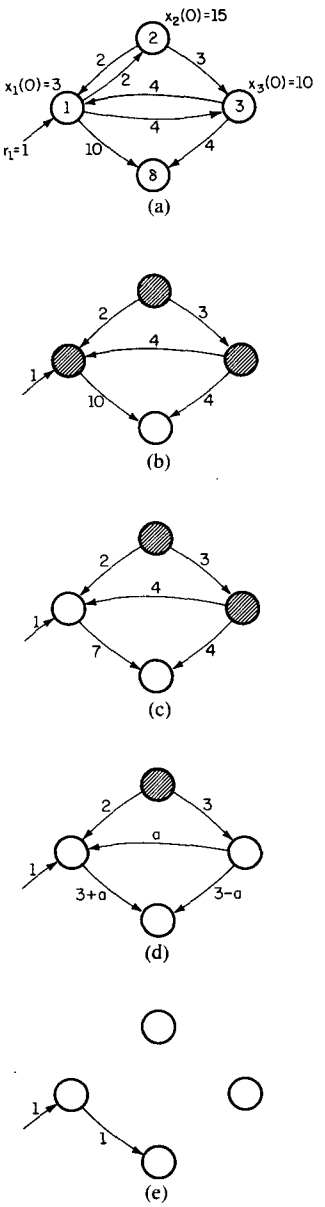


Fig. 1. Illustration of finding an optimal control by relaxing a constant control. (a) Capacity assignment and network demand. (b) Initial flow found by solving problem $\bar{P}$. The flow will be used until node 1 empties at time $t = 1$. (c) Flow obtained by reducing flow out of node 1. This flow will be used until node 3 empties at time $t = 2$. (d) Flow obtained by reducing flow out of node 3, and hence also out of node 1. Any choice of a with $0 \leqslant a \leqslant 3$ is possible. This flow will be used until node 2 empties at time $t = 3$. (e) Flow obtained as consequence of node 2 emptying. This flow will be used from time $t = 3$ onwards.

when the control $\tilde{u}^f$ is used. This suggests that a good initial flow f can be found by solving the problem $\bar{P}$ defined as follows:

$$(\bar{P}) \qquad \min \{\bar{D}(f) : f \in K\}$$

where

$$\bar{D}(f) = \sum_{i \in N-\delta} d(Bf_i + r_i, x_i(0))$$

and

$$K = \{f : 0 \leqslant f \leqslant C,$$

$$f_{i\delta} = C_{i\delta} \qquad \text{for all } i \text{ such that } (i, \delta) \in L$$

$$f_{\delta i} = 0 \qquad \text{for all } i \text{ such that } (\delta, i) \in L\}.$$

Reasonable choices for the function g are $\exp(-\alpha t)$ for some constant $\alpha \geq 0$ or $p_T(t)$ where $p_T(t) = 1$ for $0 \leq t \leq T$ and $p_T(t) = 0$ otherwise. Reference [2] deals only with the case $g(t) = 1$ for all t.

We will consider functions d more general than d of the form (2.2). We say that d is a *node cost function* if it is a function

$$d : (-\infty, \infty) \times [0, +\infty) \to (-\infty, +\infty]$$

having the following four properties.

Property 1: $d(\sigma, \theta) < +\infty$ if ($\sigma < 0$ and $\theta > 0$) or if ($\sigma = \theta = 0$).

Property 2: $d(\sigma, \theta)$ is convex lower semicontinuous in σ for each θ fixed.

Let $d'_+(\sigma, \theta)$ denote the right-hand derivative of $d(\sigma, \theta)$ *with respect to* σ.

Property 3: If $\sigma < 0$ and $d(\sigma, \theta) < \infty$, then

$$d'_+(\sigma, \theta) = \Phi(-\theta/\sigma)$$

where Φ is a nondecreasing real valued function on $[0, \infty]$. We call Φ the *scale* function of d.

Property 4: If $\sigma \geq 0$ and $d(\sigma, \theta) < \infty$, then

$$d'_+(\sigma, \theta) \geq \lim_{t \to \infty} \Phi(t).$$

Theorem 1: Let d be given by (2.2) for some function g satisfying (2.1). Then d is a node cost function with

$$\Phi(\tau) = \int_0^\tau t g(t)\, dt.$$

The proof can be found in Appendix I. We say that a node cost function d having the form (2.2) admits a gravity function g. Two node cost functions which do not admit a gravity function are

$$d(\sigma, \theta) = \begin{cases} \theta \log(-\sigma), & \text{if } \sigma < 0 \\ 0, & \text{if } \sigma = \theta = 0 \\ \infty, & \text{otherwise} \end{cases} \tag{2.3}$$

and

$$d(\sigma, \theta) = \begin{cases} \dfrac{\sigma^2}{2\theta}, & \text{if } \theta > 0 \\ 0, & \text{if } \sigma = \theta = 0 \\ \infty, & \sigma \neq 0,\ \theta = 0. \end{cases} \tag{2.4}$$

We can generate the function of (2.3) [respectively, partially generate the function of (2.4)] by using the gravity function $(t + \epsilon)^{-1}$ [respectively, $(t + \epsilon)^{-3}$)], subtracting a function of ϵ and θ from $d(\sigma, \theta)$, and then letting $\epsilon \to 0$. Direct use of t^{-1} or t^{-3} as gravity functions is inappropriate.

Henceforth we will assume that the function d in the definition of problem $\bar{P}$ is a node cost function. Letting Φ denote the scale function of d, we define F^Φ by

$$F^\Phi = \{t > 0 : \Phi(t - \epsilon) = \Phi(t + \epsilon) \quad \text{for some } \epsilon > 0\}.$$

If d admits a gravity function g, then Theorem 1 implies that F^Φ is a subset of $\{t : g(t) = 0\}$, so in particular, F^Φ is empty if $g(t) > 0$ for all t. F^Φ is also empty for the two examples (2.3) and (2.4).

We will also assume that there exists a vector f in K such that $\tilde{D}(f)$ is finite. Since $d(\cdot, x_i(0))$, and hence also $\tilde{D}$, is lower semicontinuous, this implies that there exists an optimal solution for problem $\bar{P}$.

Theorem 2: Suppose f is an optimal solution for problem $\bar{P}$. Then any relaxation $\tilde{u}^f$ of the constant control u^f minimizes $\Sigma_{i \in N-\delta} x_i(t)$ for each t not in F^Φ.

Theorem 2 is proved in Appendix II, and the following corollary is immediate.

Corollary: If κ is a nonnegative function on $(0, \infty)$ such that $\kappa(t) = 0$ for $t \in F^\Phi$, then any relaxation $\tilde{u}^f$ of an optimal solution f for problem $\bar{P}$ minimizes

$$D(u) = \int_0^\infty \kappa(t) \sum_{i \in N-\delta} x_i(t)\, dt \tag{2.5}$$

over all admissible controls. For example, κ could be the gravity function of d.

III. Iterative Methods to Solve Initial Flow Problems

A. Bertsekas' Projected Descent Direction Algorithm

In this section, we shall discuss an iterative descent algorithm that is used to solve the following problem, of which $\bar{P}$ is a special case,

$$(\tilde{P}) \qquad \min \{\tilde{D}(f) : C^- \leq f \leq C^+\}$$

where

$$\tilde{D}(f) = \sum_{i \in N-\delta} \tilde{d}_i(Bf_i),$$

$\tilde{d}_i(\cdot)$ is convex for all $i \in N - \delta$, $f = (f_e : e \in \tilde{L})$, $\tilde{L}$ is some subset of L, the vectors C^- and C^+ are called the *upper* and *lower* capacities of $\tilde{L}$, and $C^- \leq C^+$. We will assume that $\tilde{D}$ is continuously differentiable. We use this more general problem to accommodate modifications of $\bar{P}$ used later in Section III. This problem involves only *simple* constraints in the sense of Bertsekas [5, Sect. 1.5] so that Bertsekas' projected descent direction algorithm [5, Sect. 1.5] is applicable under appropriate conditions. These conditions will be given at the end of this subsection, after some definitions and the algorithm are presented.

Let $[\cdot]^\#$ represent projection onto $\{f : C^- \leq f \leq C^+\}$, i.e., for all $e \in \tilde{L}$

$$[f]^\#_e = \max\{\min\{C^+_e, f_e\}, C^-_e\}.$$

Let $f(n)$ be the vector produced at the nth step of Bertsekas' algorithm for $n \geq 1$, and let $f(0)$ be the initial value. Denote

$$\epsilon(n) = \min\{\epsilon, \|f(n) - [f(n) - \nabla\tilde{D}(f(n))]^\#\|\}$$

where ϵ is a fixed strictly positive scalar (typically small), $\nabla\tilde{D}(f(n))$ is the gradient of $\tilde{D}$ at $f(n)$, and $\|\cdot\|$ is the Euclidean norm. Let

$$I(n) = \left\{ e \in \tilde{L} : \begin{array}{l} f(n)_e \leq C^-_e + \epsilon(n) \text{ and } \nabla\tilde{D}(f(n))_e > 0 \text{ or} \\ f(n)_e \geq C^+_e - \epsilon(n) \text{ and } \nabla\tilde{D}(f(n))_e < 0 \end{array} \right.$$

and $W(n) = \tilde{L} - I(n)$. Roughly speaking, $\epsilon(n)$ is an *a priori* estimate of how much $f(n)$ will change during the n + 1st iteration and $I(n)$ is an *a priori* estimate of the set of coordinates for which constraints will restrict the change in $f(n)$.

Let σ, β, and γ be fixed, strictly positive scalars where $\sigma < 1/2$ and $\beta < 1$. Let

$$\alpha(m) = \gamma\beta^m$$

$$f(n, \alpha) = [f(n) - \alpha p(n)]^\#$$

$$p(n) = G(n)\nabla\tilde{D}(f(n))$$

$$G(n) = \operatorname{diag}(h(n), \zeta(n))$$

where $h(n)$ is a diagonal matrix corresponding to the coordinates in $I(n)$ and $\zeta(n)$ is a matrix corresponding to the

coordinates in $W(n)$. Typically, $G(n)$ will be chosen as an approximation to the inverse Hessian of $\tilde{D}$ evaluated at $f(n)$, as suggested by the classical method of Newton. Choices for $G(n)$ are explored in Section III-C.

n + 1st Iteration of Bertsekas' Projected Descent Algorithm:

$$f(n+1)=f(n,\ \alpha_n)$$

where

$$\alpha_n=\alpha(m_n) \tag{3.1}$$

and m_n is the smallest nonnegative integer m that passes the following test.

Step Size Test (SST):

$$\tilde{D}(f(n))-\tilde{D}(f(n,\ \alpha(m)))$$

$$\geqslant \sigma\left[\begin{array}{l}\alpha(m)\displaystyle\sum_{e\in W(n)}\nabla\tilde{D}(f(n))_e p(n)_e \\ +\displaystyle\sum_{e\in I(n)}\nabla\tilde{D}(f(n))_e[f(n)-f(n,\ \alpha(m))]_e\end{array}\right].$$

Roughly speaking, the SST requires that the actual amount of descent is at least a fraction σ of the amount of descent predicted by first derivative information, the step size, and the constraints.

Equation (3.1) and SST is called the *Armijo-like* rule since it is similar to the *Armijo* rule for finding step sizes [6]. A variation of the Armijo-like rule is the *Markov–Armijo-like* rule, which uses the previous step size as the initial step size tested. The Markov–Armijo-like step for the nth iteration is

$$\alpha_n=\alpha(m_n) \qquad \text{for all } n \geqslant 0$$

where $m_{-1}=0$ and m_n is generated as follows. Initialize m to equal m_{n-1}. If the SST is true for m, then we decrease m until either the SST is false or $m=-1$, then set $m_n=m+1$. Otherwise, we increase m until the SST is true, then set $m_n=m$.

The Markov–Armijo-like step size rule should work quite well if α_n is fairly constant over n. The rule should also work better than the Armijo-like rule if the initial step size γ of the Armijo-like rule is poorly chosen. We believe that the Markov–Armijo-like rule will not perform much worse than the Armijo-like rule because if, for a particular instance, both rules generate the same step sizes (this happens in unconstrained problems with convex cost since the set of m satisfying the SST has the form $\{m:m \geqslant \hat{m}\}$), then the Markov–Armijo-like rule will use not more than approximately twice as many SST's as the Armijo-like rule will use.

The following are conditions sufficient to ensure that the sequence $\{f(n)\}$ converges to a local minimum, and since $\tilde{P}$ has a convex cost, the local minimum is also a global one.

Condition A: Given any bounded subset S of $\mathbb{R}^L$, there exists a scalar Y (depending on S) such that

$$\|\nabla\tilde{D}(x)-\nabla\tilde{D}(y)\| \leqslant Y\|x-y\| \qquad \text{for all } x,\ y \text{ in } S.$$

Condition B: There exist positive scalars λ_1 and λ_2 such that for all $n \geqslant 0$

$$\lambda_1\|z\|^2 \leqslant z^T G(n) z \leqslant \lambda_2\|z\|^2 \qquad \text{for all } z$$

where z^T is the transpose of the vector z.

B. Approximation Techniques for Problems Which Have Nondifferentiable Cost Functions or Which Do Not Satisfy Condition A

The cost function of $\bar{P}$ need not be continuously differentiable nor satisfy Condition A. An example where $\bar{D}$ may not be continuously differentiable is when d corresponds to the gravity function $e^{-\alpha t}$ because $d(\sigma, 0)$ is then not differentiable with respect to σ at zero. An example where $\bar{D}$ may not satisfy Condition A is when the node cost function is the function of (2.3).

One possible remedy is to replace the node cost function $d(\sigma, \theta)$ in $\bar{P}$ with $d(\sigma, \max(\theta, \omega))$ where ω is some small positive constant. We call this new problem P^ω. Note that if the node cost function of $\bar{P}$ admits one of the gravity functions $e^{-\alpha t}$ or p_T or if it is the function of (2.4), then P^ω has a continuously differentiable cost and satisfies Condition A. Also note that if $\min\{x_i(0): i \in N-\delta\} \geqslant \omega$, then P^ω is equivalent to $\bar{P}$. Optimal solutions of P^ω lead to nearly optimal controls in the following sense [7]. Suppose for each $\omega \geqslant 0$ there is a finite optimal solution f^ω for P^ω. Let $x^\omega(t)$ be a state trajectory corresponding to f^ω and demand $(x(0), r)$. If $t \in F^\phi$, then

$$\sum_{i\in N-\delta} x_i^\omega(t) \leqslant \sum_{i\in N-\delta} x_i^0(t)+|N|\omega$$

and if, in addition, $t \geqslant t_1+t_2\omega$, then

$$\sum_{i\in N-\delta} x_i^\omega(t)=\sum_{i\in N-\delta} x_i^0(t)$$

where t_1 and t_2 are dependent on (N, L), $(x(0), r)$, and C only.

Another possible way to make cost functions differentiable and satisfy Condition A is to use Bertsekas' method of multipliers for nondifferentiable and ill-conditioned problems [5, Sect. 3.3]. This method applies nicely to the cost functions of (2.3) and (2.4).

C. Descent Direction Methods

We next describe methods to produce a descent direction $-G(n)\nabla D(f(n))$ such that $G(n)$ satisfies Condition B. One simple method is to let $G(n)$ equal the density matrix. The descent direction will then correspond to the direction of *steepest* descent of $\tilde{D}$ and leads to linear convergence. A descent direction that leads to superlinear convergence is the *Newton* descent direction, i.e., $G(n)$ is the inverse of the Hessian matrix of $\tilde{D}$ evaluated at $f(n)$. Computing Newton descent direction takes $O(|N|^3)$ time, which may be too time consuming. Intuitively, we expect that the closer $-G(n)\nabla\tilde{D}f(n))$ approximates a Newton descent direction, the better the descent direction will be, but at the expense of increased time to compute $-G(n)\nabla\tilde{D}(f(n))$. In this subsection, we will present descent directions that are Newton-like, but can be computed in a reasonable amount of time.

Assume that, for all $i \in N-\delta$, $\tilde{d}_i(\cdot)$ has first and second derivatives. Then the $[(a, b), (c, d)]$ element of the Hessian matrix of $\tilde{D}$ evaluated at f is

$$Q_{(a,b)(c,d)}(f)=\mu_a(f)I_{[a=c]}+\mu_b(f)I_{[b=d]}-\mu_a(f)I_{[a=d]}-\mu_c(f)I_{[b=c]}$$

where

$$I_A=\begin{array}{ll}1 & \text{if } A \text{ is true}\\ 0 & \text{otherwise}\end{array}$$

where $\mu_i(f)$ equals the second derivative of $\tilde{d}_i(\cdot)$ at Bf_i. In order for our descent directions to satisfy Condition B, we also assume that for some constants π_1 and π_2, $0 \leqslant \pi_1 \leqslant \mu_i(f) \leqslant \pi_2$ for all i in $N-\delta$. Such a π_1 always exists since the $\tilde{d}_i$ are convex.

Remark: If the functions $\tilde{d}_i(\cdot)$ and $\mu_i(\cdot)$ do not satisfy the assumptions above, we modify Q as follows. In case $\tilde{d}_i(\cdot)$ does not have a second derivative at Bf_i, we simply assign an arbitrary positive value to $\mu_i(f)$. In case $\mu_i(f)$ is not bounded between π_1 and π_2, we replace it with $\min\{\pi_2, \max\{\pi_1, \mu_i(f)\}\}$.

For the next two methods, we will assume $\pi_1 > 0$, which will be used to show that Condition B is satisfied.

Example (Diagonal Approximation Method): In this method, we approximate Q by its diagonal part and use the inverse of the approximation for $G(n)$. Thus, for each link e, we have $[G(n)\nabla\tilde{D}(f(n))]_e = \nabla\tilde{D}(\tilde{f}(\tilde{n}))_e/\tilde{Q}(\tilde{f}(\tilde{n}))_{ee}$. Since $0 < 2\pi_1 \leqslant Q(f(n))_{ee} \leqslant 2\pi_2$, $G(n)$ satisfies Condition B.

In the next two examples, we will discuss computation of the coordinates of the descent vector indexed by links in $W(n)$. These links correspond to that part of matrix $G(n)$ that does not have to be diagonal. Let $\hat{Q}$ be the submatrix of Q corresponding to the coordinates in $W(n)$.

Example (Block Diagonal Approximation or Partitioning Method): In this method, we approximate $\hat{Q}(f(n))$ by a *block* diagonal matrix $Q'(n)$ and use the inverse of $Q'(n)$ for $\zeta(n)$. To form $Q'(n)$, we first assign each link to one of its two end nodes. Without loss of generality, let $\{1, 2, \cdots, y\}$ be the set of nodes that have links assigned to it. Let $Q'(n)$ be the block diagonal matrix such that

$$Q'_{uv}(n) = \begin{cases} \hat{Q}_{uv}(f(n)), & \text{if links } u \text{ and } v \text{ are assigned to the same node} \\ 0, & \text{otherwise.} \end{cases}$$

Thus, $Q'(n)$ has the form diag $(H_1(n), H_2(n), \cdots, H_y(n))$ where $H_i(n)$ is the submatrix of $\hat{Q}(f(n))$ corresponding to the links assigned to node i.

In order to show that the resulting matrix $G(n)$ satisfies Condition B, it is sufficient to show that matrix $H_i(n)$ is uniformly bounded above and away from zero, i.e.,

$$\tilde{\pi}_1\|z\|^2 \leqslant z^T H_i(n) z \leqslant \tilde{\pi}_2\|z\|^2 \qquad \text{for all } z \tag{3.1}$$

for some $0 < \tilde{\pi}_1 < \tilde{\pi}_2$. To do this, we note that we can arrange it so that there exists at most one link incident to the same pair of nodes because if there exist two links incident to the same pair of nodes, we convert these two links into a single link by deleting one of the links and assigning a new lower (respectively, upper) capacity to the remaining link that is equal to its old lower (respectively, upper) capacity minus (respectively, plus) the upper (respectively, lower) capacity of the deleted link.

Let the links assigned to node i be $\{(j_1, i), (j_2, i), \cdots, (j_p, i), (i, j_{p+1}), (i, j_{p+2}), \cdots, (i, j_{p+m})\}$. From our assumption that there is at most one link between any pair of nodes, the j_k are all distinct. Then we can write

$$H(n)_i = \mu_i(f(n))A_i(n) + \text{diag}\,(\mu_{j_1}(f(n)), \cdots, \mu_{j_{p+m}}(f(n))) \tag{3.2}$$

where

$$A_i(n) = \begin{bmatrix} E_{pp} & -E_{pm} \\ -E_{mp} & E_{mm} \end{bmatrix}$$

and E_{pm} is a $p \times m$ matrix of ones. By straightforward computation,

$$z^T H_i(n) z = \mu_i(f(n))\left[\sum_{k=1}^{p} z_k - \sum_{k=p+1}^{p+m} z_k\right]^2 + \sum_{k=1}^{p+m} \mu_{j_k}(f(n)) z_k^2.$$

From the boundedness of $\mu_i(f(n))$ and the fact that $1 \leqslant p + m \leqslant |\bar{L}|$, the following inequalities hold, which implies (3.1) for $\tilde{\pi}_1 = \pi_1$ and $\tilde{\pi}_2 = \pi_2(1 + |\bar{L}|)$,

$$0 \leqslant \mu_i(f(n))\left[\sum_{k=1}^{p} z_k - \sum_{k=p+1}^{p+m} z_k\right]^2 \leqslant \pi_2|\bar{L}|\|z\|^2$$

$$\pi_1\|z\|^2 \leqslant \sum_{k=1}^{p+m} \mu_{j_k}(f(n)) z_k^2 \leqslant \pi_2\|z\|^2.$$

Thus, Condition B is satisfied.

By straightforward calculation, the (i, j)th coordinate of the descent direction for a link (i, j) of $W(n)$ at the nth iteration of the algorithm of Section III-A is

$$-\mu_j(f(n))^{-1}[\nu_i\mu_i(f(n)) + \nabla\tilde{D}(f(n))_{(i,j)}] \qquad \text{if } (i, j) \in M_i(n)$$

$$-\mu_j(f(n))^{-1}[-\nu_i\mu_i(f(n)) + \nabla\tilde{D}(f(n))_{(i,j)}] \qquad \text{if } (j, i) \in M_i(n)$$

where $M_i(n)$ is the subset of $W(n)$ assigned to node i and

$$\nu_i = -\frac{\sum_{(i,p)\in M_i(n)} \mu_p(f(n))^{-1}\nabla\tilde{D}(f(n))_{(i,p)} - \sum_{(p,i)\in M_i(n)} \mu_p(f(n))^{-1}\nabla\tilde{D}(f(n))_{(p,i)}}{1 + \mu_i(f(n))\left[\sum_{(p,i)\in M_i(n)} \mu_p(f(n))^{-1} + \sum_{(i,p)\in M_i(n)} \mu_p(f(n))^{-1}\right]}.$$

We next give a heuristic rule for assigning links to nodes in an attempt to make the approximation $Q'(n)$ of $\hat{Q}(f(n))$ as close as possible.

Partition Rule: Let (i, j) be assigned to node i at iteration n if

$$\mu_i(f(n))\sqrt{\Omega_i - 1} \geqslant \mu_j(f(n))\sqrt{\Omega_j - 1}$$

and let (i, j) be assigned to node j otherwise where Ω_k is the number of links in $W(n)$ incident on node k.

The motivation behind this rule can be observed through the following situation. Suppose we have a network where all nodes are a neighbor of node a or node b, but not both, as in Fig. 2. Furthermore, assume that links incident to node a but not node b are assigned to node a, and that links incident to node b but not node a are assigned to node b. Let $k \in \{a, b\}$ and let Q'_k be $Q'(n)$ when (a, b) is assigned to node k. To measure how close an approximation a matrix H is to $\hat{Q}(f(n))$, we use the distance

$$\Delta(H, \hat{Q}(f(n))) = \sup\{\|(H - \hat{Q}(n))y\|/\|y\| : y \neq 0\}.$$

By straightforward computation,

$$\Delta(\hat{Q}(f(n)), Q'_k) = \mu_k(f(n))\sqrt{\Omega_k - 1}.$$

Therefore, the Partition Rule will give us the best approximation of $\hat{Q}(f(n))$.

Remark: Suppose the cost $\tilde{D}$ is such that, given f, computing $\nabla\tilde{D}(f)$, $(\mu_i(f): i \in N - \delta)$, and $\tilde{D}(f)$ takes $O(|N|^2)$ time. An example of such a $\tilde{D}$ is $\bar{D}$ when the node cost function is either $d(p_T(t), \cdot, \cdot)$ or the function of (2.4). Also, suppose that we apply the iterative descent algorithm of Section III-A where the descent direction is either the diagonal or block diagonal approximation descent direction to solve $\tilde{P}$. Then it can be shown that computing a descent direction, evaluating an SST, or updating an initial flow takes $O(|N|^2)$ time. Hence, the nth iteration of the algorithm will take $O(m_n|N|^2)$ time where m_n is the number of SST evaluations

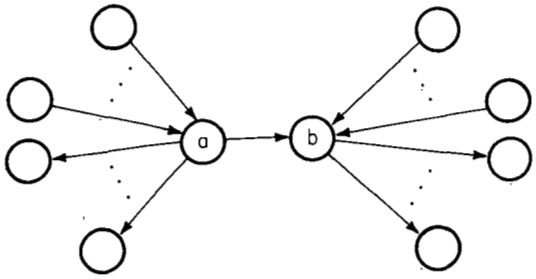


Fig. 2. The network used to motivate the partition rule.

needed to find an acceptable step size at the nth iteration. Since the descent direction is Newton-like, the acceptable step size should be close to one, and m_n should be less than some fixed constant for most $|N|$.

Example (Scaled Conjugate Gradient Method): In this example, we use the *scaled conjugate gradient method* that was used in [8]. Let $\tilde{Q}(n) = \tilde{Q}(f(n)) + \epsilon I$ where $\epsilon > 0$ and I is the identity matrix. Let $S(n)$ be a symmetric matrix which has eigenvalues bounded above and away from zero. Let $\nabla \tilde{D}(f(n))_E = (\nabla \tilde{D}(f(n))_e : e \in E)$ for E a subset of $\tilde{L}$. Let the positive integer sequence $\{c(n)\}$ be such that $0 < c(n) < |W(n)|$. Then the descent direction vector corresponding to the links in $W(n)$ is $z(c(n))$ where z is computed iteratively as follows. Let $z(0) = 0$, $y(0) = -S(n)w(0)$.

$$z(m) = z(m-1) + \xi(m-1)y(m-1)$$

$$y(m) = -S(n)w(m) + \nu(m)y(m-1)$$

$$w(m) = \tilde{Q}(n)z(m) + \nabla \tilde{D}(f(n))_{W(n)}$$

$$\xi(m) = \frac{w(m)^T S(n) w(m)}{y(m)^T S(n) y(m)}$$

$$\nu(m) = \frac{w(m)^T S(n) w(m)}{w(m-1)^T S(n) w(m-1)}.$$

Note that if the Hessian matrix of $\tilde{D}$ exists and is nonsingular at $f(n)$ and $\epsilon = 0$, then $z(|W(n)| - 1)$ is the Newton descent direction.

IV. Results of Computer Experiments

Computer experiments done to compare the methods presented in Sections II and III are described in [7]. In the experiments, the objective was to find a routing strategy to minimize $D(u)$ in (2.5) when the weight function κ is p_T where T was chosen about ten times larger than the minimum time to empty the networks. This was done by applying the iterative descent algorithm of Section III-A to find an acceptable feasible vector for $\bar{P}$. The criterion for a feasible vector f of $\bar{P}$ to be acceptable was that

$$\sum_{i \in N-\delta} d(Bf_i, x_i(0)) \leqslant D(1+q)$$

for some small positive q, d corresponds to gravity function κ, and D is the minimum cost possible. Note that if $\tilde{u}^f$ is a flow relaxation of u^f, then $D(\tilde{u}^f) \leqslant D(1+q)$. The implementation of the algorithm was done using the Pascal programming language on a Digital Equipment Corporation LSI-11 microcomputer. Extensive numerical results are reported in [7]; here we present a brief summary.

Experimental results comparing the two step size rules show that the Markov–Armijo-like rule never used more than 128 percent and at one time used only 12 percent of the SST's used by the Armijo-like rule. Bertsekas and an anonymous referee suggested the following modification for the Markov–Armijo-like step size rule. When computing the integer m_n, use the same procedure as in the Armijo-like or Markov–Armijo-like rules, except let m_n be the value of m that gives the biggest decrease in cost. We have not conducted any experiments with this rule, but we think it will work quite well.

To compare the performance of the algorithm for various descent direction methods and for various node cost functions, we measured how much CPU time was needed by the algorithm to compute an acceptable initial flow. We tried the following descent direction methods: steepest descent, diagonal approximation, block diagonal approximation, partial scaled conjugate gradient, and the Newton descent. The experimental results showed that the block diagonal and partial scaled conjugate gradient method, for approximately $|W(n)|/2$ conjugate gradient iterations, performed best. We tried four node cost functions for $\bar{P}$: using gravity functions $\kappa(t)$ and $\exp(-(0.1)t)$, and the examples of (2.3) and (2.4). The algorithm worked best with the quadratic cost of (2.4).

If the cost $\bar{P}$ was not continuously differentiable or Condition A was not satisfied, we substituted P^ω for $\bar{P}$ or applied Bertsekas' multiplier method whenever possible. The experimental results did not indicate which of the two techniques would lead to faster computation times for the algorithm.

V. Conclusion

In this paper, conditions are given on the initial flow problem $\bar{P}$ which ensure that its solution leads to controls which minimize the amount of traffic in the network at some set of times. We used Bertsekas' iterative descent algorithm to solve $\bar{P}$. An alternative to the Armijo-like rule, called the Markov–Armijo-like rule, was presented and it seems to perform at times better and never perform much worse than the Armijo-like rule.

Conditions that are sufficient for the algorithm to converge to an optimal solution are given. We presented techniques to compute descent directions and to modify problem $\bar{P}$ so that the conditions are satisfied. Among the descent direction methods is the block diagonal approximation, which seems like a reasonable method. We also note that the choice of the node cost function for $\bar{P}$ can affect the rate of convergence of the algorithm.

Appendix I

Proof of Theorem 1

We will only prove that d has Property 2. The proof that d has the other three properties is left to the reader. $d(\sigma, \theta)$ is convex in σ because $[\theta + t\sigma]_+$ is convex in σ. Next, suppose that σ_n converges to σ. Then $g(t)[\theta + \sigma_n t]_+$ is nonnegative and converges to $g(t)[\theta + \sigma t]_+$ so by Fatou's lemma,

$$\int_0^\infty g(t)[\theta + \sigma t]_+ \, dt \leqslant \liminf_{n \to \infty} \int_0^\infty f(t)[\theta + \sigma_n t]_+ \, dt$$

which, by definition, means that $d(\sigma, \theta)$ is a lower semicontinuous function of σ. Thus, d has Property 2. □

Appendix II

Proof of Theorem 2

We prove the theorem using the following set of claims and definitions.

Definition: A *path* with length $q \geqslant 0$ is a sequence of unrepeated nodes $(n_0, n_1, \cdots, n_q)$ with links between them. If f is in K, then the path is said to be *unsaturated* for f if for each i with $0 \leqslant i \leqslant q - 1$,

$$C_{n_i, n_{i+1}} - f_{n_i n_{i+1}} + f_{n_{i+1}, n_i} > 0.$$

By default, a path with a single node is unsaturated for f.

Let $d'_-(\sigma, \theta)$ be the left-hand derivative with respect to σ

evaluated at σ. Note that since $\bar{D}(f) < \infty$ and $d(\cdot, x_i(0))$ is convex, $d'_-(Bf_i + r_i, x_i(0))$ and $d'_+(Bf_i + r_i, x_i(0))$ are well defined for each i in $N - \delta$.

Claim 1: Let p be an unsaturated path from u to v where $u \neq v$. Then the destination is not on the path and

$$d'_-(Bf_u + r_u, x_u(0)) \leqslant d'_+(Bf_v + r_v, x_v(0)). \quad \text{(A.2.1)}$$

Proof: Since f is a solution, f is in K. Therefore, $f_{i\delta} = C_{i\delta}$ and $f_{\delta i} = 0$ for all i in $N - \delta$, and δ cannot be part of p. Since p is unsaturated, there exists a vector w where

$$w_{ij} = \begin{cases} 1 & \text{if } (i, j) \text{ is part of path } p \text{ and } f_{ij} < C_{ij} \\ -1 & \text{if } (j, i) \text{ is part of path } p \text{ and } f_{ji} = C_{ji} \\ 0 & \text{otherwise} \end{cases}$$

and $f + \alpha w \in K$ for some $\alpha > 0$ sufficiently small. By the convexity of $\bar{D}$ and the optimality of f, the following limit exists and

$$\lim_{\epsilon \to 0: \epsilon > 0} \frac{1}{\epsilon} [\bar{D}(f + \epsilon w) - \bar{D}(f)] \geqslant 0. \quad \text{(A.2.2)}$$

The vector w has the property that

$$Bw_i = \begin{cases} 1 & \text{if } i = v \\ -1 & \text{if } i = u \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the limit on the left-hand side of (A.2.2) equals

$$\lim_{\epsilon \to 0: \epsilon > 0} \frac{1}{\epsilon} [d(Bf_u + r_u - \epsilon, x_u(0)) - d(Bf_u + r_u, x_u(0)) - (d(Bf_v + r_v + \epsilon, x_v(0)) - d(Bf_v + r_v, x_v(0)))]$$
$$= d'_+(Bf_v + r_v, x_v(0)) - d'_-(Bf_u + r_u, x_u(0))$$

which completes our proof of Claim 1. □

Fix, for the remainder of this Appendix, a value $t > 0$ not in F^Φ.

Claim 2: Suppose there is an unsaturated path from u to v and that $x_u(0) + t(Bf_u + r_u) > 0$. Then $x_v(0) + t(Bf_v + r_v) \geqslant 0$.

Proof: We can restrict attention to the case that $Bf_v + r_v < 0$. Then by Claim 1 and Property 3,

$$d'_-(Bf_u + r_u, x_u(0)) \leqslant d'_+(Bf_v + r_v, x_v(0)) = \Phi\left(\frac{x_v(0)}{-Bf_v - r_v}\right). \quad \text{(A.2.3)}$$

Now choose ζ with $\zeta > t$ so that $x_u(0) + \zeta(Bf_u + r_u) > 0$. Then if $x_u(0) > 0$, we have $Bf_u + r_u > -x_u(0)/\zeta$ so Properties 1.2, and 3 yield

$$d'_-(Bf_u + r_u, x_u(0)) \geqslant d'_+(-x_u(0)/\zeta, x_u(0)) = \Phi(\zeta). \quad \text{(A.2.4)}$$

On the other hand, if $x_u(0) = 0$ (so then $Bf_u + r_u > 0$), we have by the convexity of $d(\cdot, \theta)$ and the assumption $d(0, 0)$ is finite that

$$d((Bf_u + r_u)/2, x_u(0)) \leqslant d(0, x_u(0)) + d(Bf_u + r_u, x_u(0)) < +\infty.$$

Thus, Property 4 and the monotonicity of Φ yield

$$d'_-(Bf_u + r_u, x_u(0)) \geqslant d'_+((Bf_u + r_u)/2, x_u(0)) \geqslant \lim_{u \to \infty} \Phi(u) \geqslant \Phi(\zeta). \quad \text{(A.2.5)}$$

Combining (A.2.3) and either (A.2.4) or (A.2.5), we have

$$\Phi(\zeta) \leqslant \Phi\left(\frac{x_v(0)}{-Bf_v + r_v}\right).$$

This and the fact that $\zeta > t$, Φ is monotone and $t \notin F^\Phi$ implies that $x_v(0)/(-Bf_v + r_v) \geqslant t$, which establishes Claim 2. □

Let $x_A = \Sigma_{i \in A} x_i$ for any subset A of N and let $C_{AB} = \Sigma_{(i,j): i \in A \text{ and } j \in B} C_{ij}$ for subsets A and B of N. Let N^+ denote the set of nodes i in $N - \delta$ such that $x_i(0) + t(Bf_i + r_i) > 0$ and let N^- denote the set of nodes in $N - \delta$ such that $x_i(0) + t(Bf_i + r_i) < 0$. Let R be the set of all nodes that can be reached from a node in N^+ by way of an unsaturated path. By Claim 1, R does not include δ, and by Claim 2, R does not intersect N^-. By its definition, R clearly contains N^+ and

$$f_{N-R,R} = 0 \text{ and } f_{R,N-R} = C_{R,N-R}.$$

Let x denote the trajectory for an arbitrary admissible control and let x^* denote the trajectory for the control $\bar{u}^f$. Then

$$x_{N-\delta}(t) \geqslant x_R(t) \geqslant x_R(0) + \int_0^t (r_R - C_{R,N-R})\, ds$$
$$= x_R(0) + t(r_R - C_{R,N-R})$$
$$= x_R(0) + t(r_R + f_{n-R,R} - f_{R,N-R})$$
$$= \sum_{i \in R} [x_i(0) + t(Bf_i + r_i)]$$
$$= \sum_{i \in N-\delta} [x_i(0) + t(Bf_i + r_i)]_+. \quad \text{(A.2.6)}$$

As discussed in Section II, we have $[x_i(0) + t(Bf_i + r_i)]_+ \geqslant x_i(t)$, whence from (A.2.6) we deduce that $x_{N-\delta}(t) \geqslant x_{N-\delta}(t)$. This completes the proof of Theorem 2. □

REFERENCES

[1] A. Segall, "The modeling of adaptive routing in data communication networks," *IEEE Trans. Commun.*, vol. COM-25, pp. 85–95, Jan. 1977.

[2] B. Hajek and R. G. Ogier, "Optimal dynamic routing in communication networks with continuous traffic," *Networks*, vol. 14, pp. 457–487, 1984.

[3] G. I. Stassinopoulos and P. Konstantopoulos, "Optimal congestion control in single destination networks," *IEEE Trans. Commun.*, vol. COM-33, pp. 792–800, Aug. 1985.

[4] A. Feit, "Dynamic multicommodity flow schedules," Ph.D. dissertation, Naval Postgrad. School, Monterey, CA, Dec. 1981.

[5] D. P. Bertsekas, *Constrained Optimization and Lagrange Multipler Methods.* New York: Academic, 1982.

[6] J. Armijo, "Minimization of functions having continuous partial derivatives," *Pacific J. Math.*, vol. 16, pp. 1–3, 1966.

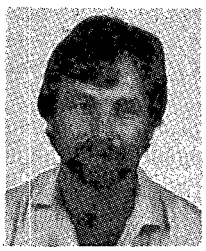
[7] G. H. Sasaki, "Optimal dynamic routing by iterative methods," M.S. thesis, Coord. Sci. Lab., Univ. Illinois, Urbana-Champaign, Rep. T-143, July 1984.

[8] D. P. Bertsekas and E. M. Gafni, "Projected Newton methods and optimization of multicommodity flows," *IEEE Trans. Automat. Contr.*, vol. AC-28, pp. 1090–1096, Dec. 1983.

★

Galen Sasaki received the B.S. degree in electrical engineering from the University of Hawaii, Manoa, in 1981, and the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois, Urbana-Champaign, in 1984 and 1987, respectively.

Presently, he is an Assistant Professor in the Department of Electrical and Computer Engineering, University of Texas, Austin.



Bruce Hajek (M'79-SM'84) is currently a Professor in the Department of Electrical and Computer Engineering and in the Coordinated Science Laboratory at the University of Illinois, Urbana-Champaign, where he has been since completing his graduate work in electrical engineering at Berkeley in 1979. He was a Visiting Scientist at M.I.T. (Spring of 1986) and at the Institute for Problems in Transmission of Information in Moscow (November-December 1986) under the joint National Academy of Sciences Interacademy Exchange Program. He is the Associate Editor for Communication Networks and Computer Networks of the IEEE TRANSACTIONS ON INFORMATION THEORY. His research interests include multiple-user communication theory, information theory, random fields and other stochastic processes, stochastic control, and combinational optimization.

Dr. Hajek was a winner of the 1973 USA Mathematical Olympiad, and he received the Eckman Award of the American Automatic Control Council (1982), an NSF Presidential Young Investigators Award (1984), and an Outstanding Paper Award from the IEEE TRANSACTIONS ON AUTOMATIC CONTROL (1985).