Computer Networks 66 (2014) 18–31

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Flow Deviation: 40 years of incremental flows for packets, waves, cars and tunnels

Luigi Fratta [a,*], Mario Gerla [b], Leonard Kleinrock [b]

[a] DEIB, Politecnico di Milano, Italy
[b] Comp Science Dept, UCLA, United States

ARTICLE INFO

Keywords:
Flow optimization
Multicommodity flows
Routing
Topology design
Flow Deviation algorithm
ARPANET
INTERNET
SDN
WDM
Vehicular traffic.

ABSTRACT

It was about 40 years ago that the ARPANET had surpassed 40 nodes and was rapidly growing at the rate of one node per month. The first government and commercial P/S networks were emerging. Designing an efficient topology and flow assignment strategy that could meet the requirements at lowest costs was essential in order to outperform the competition represented by circuit-switched or private-line networks. The challenge was to solve a combined flow and capacity assignment problem. Researchers at UCLA, under Len Kleinrock's leadership and DARPA support, developed a method based on multicommodity flow and non-linear programming principles called the Flow Deviation (FD) algorithm. The key underlying technique was to "deviate flows" on incrementally improving paths until the optimum was reached. In the early days, the FD algorithm was used to design and "upgrade" the growing ARPANET topology. After the ARPANET topology became stable, the FD algorithm took on a life of its own as an intuitive, efficient method to solve a variety of network flow problems in different domains, from ARPANET packets to WDM wavelengths, urban traffic cars and SDN tunnels, as reflected in the over 500 citations.

This paper reviews the key concepts and features of the FD algorithm as published in 1973, traces the various problems and scenarios to which it has been applied (most recently, vehicular traffic management) and discusses the possible use of FD in today's SDN tunnel allocation.

1. Introduction

The idea of the Flow Deviation method came from the convergence of needs and opportunities. In 1971 the ARPANET was growing beyond the size that could be designed by hand, so the need was to deliver an automated method to find the best topology layout. At the same time, the theory of networks was emerging then as an exciting new area of research, both in terms of "stochastic networks of queues" and "network flow optimization" [1,2,4]. Prior to joining UCLA, Kleinrock had already formulated and solved the capacity assignment problem in his 1962 PhD dissertation which was later republished as a book [4] in 1964. After joining UCLA, Kleinrock introduced in the engineering curriculum a course in queuing networks followed by a course in network flows. Putting together need and opportunity, two young researchers, Luigi Fratta and Mario Gerla, formulated two basic problems in the design of a packet-switched network like the ARPANET and used multicommodity flow and queuing network teachings from Kleinrock's classes to solve them. Quoting the original Flow Deviation paper "*...Two problems relevant to the design of a store-and-forward communications network (the message routing problem and the channel capacity assignment problem) are formulated and are recognized to be essentially*

* Corresponding author. Tel.: +39 02 2399 3578.
E-mail address: fratta@elet.polimi.it (L. Fratta).

http://dx.doi.org/10.1016/j.comnet.2014.04.001
1389-1286/

non-linear, unconstrained multicommodity (m.c.) flow problems. A "Flow Deviation" (FD) method for the solution of these non-linear, unconstrained flow problems is described which is quite similar to the gradient method for functions of continuous variables. Here the concept of gradient is replaced by the concept of "shortest route" flow. As in the gradient method, the application of successive flow deviations leads to local minima. Finally, two interesting applications of the FD method to the design of the ARPA Computer Network are discussed." [19].

This paper reviews the basic flow deviation (FD) procedure of assigning flows within store-and-forward communications networks so as to minimize cost and/or delay for a given topology and for given external flow requirements. Various approaches to the problem are discussed and the FD method is introduced and described. The method is applied to the routing problem in the ARPA network [19]. After this brief review of the original paper, the literature was surveyed, and several FD applications (developed after the FD paper was first introduced in 1972) to different problem domains are reported. Finally, possible uses of the FD technique in modern SDN networks are explored.

2. Multicommodity flow problem formulation

Suppose we have a collection of nodes N_i, ($i = 1, \ldots, n$), and are required to route a quantity r_{ij} of type (i, j) commodity from N_i to N_j through a given network (Fig. 1).

The multicommodity (m.c.) flow problem consists of finding the routes for all such commodities, which minimize (or maximize) a well-defined performance function (e.g., cost or delay), such that a set of constraints (e.g., channel capacity constraints) are satisfied.

The most general multicommodity problem can be expressed formally in the following way:

Given:	A network of n nodes and b arcs An $n \times n$ matrix $R = [r_{ij}]$, called the requirement matrix, whose entries are non-negative
Minimize:	(or maximize) $P(\Phi)$
Over Φ:	where Φ is the flow configuration and P is a well-defined performance function

Furthermore, Φ must satisfy the following constraints:

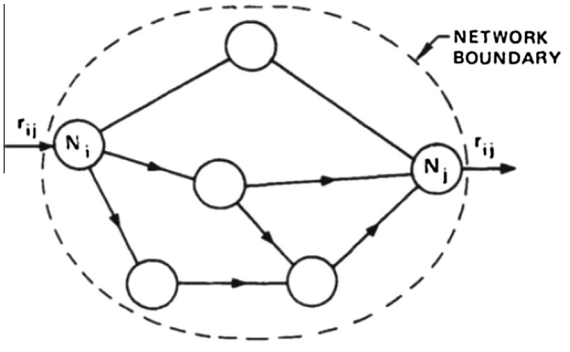


Fig. 1. Example of routing the (i, j) commodity.

1. Φ must be a multicommodity flow satisfying requirement R. For this, the following conditions must be verified:

Conservation of flows at nodes, commodity by commodity:

$$\sum_{k=1}^{n} f_{kl}^{(ij)} - \sum_{m=1}^{n} f_{lm}^{(ij)} = \left\{ \begin{array}{cc} -r_{ij} & \text{if } l = i \\ +r_{ij} & \text{if } l = j \\ 0 & \text{otherwise} \end{array} \right\} \forall i, j \tag{2.1}$$

Non-negativity of flow in directed arcs:

$$f_{kl}^{(ij)} \geqslant 0 \quad \forall \quad i, j, k, l \tag{2.2}$$

where $f_{kl}^{(ij)}$ is the portion of commodity (i, j) flowing on arc (k, l).

2. Φ must satisfy some additional constraints, different from problem to problem (e.g., capacity constraints on each channel and/or cost constraints). *Note:* If an m.c. flow problem has no additional constraints, we define it to be an unconstrained m.c. flow problem. Such a definition will be motivated in one of the following sections.

Let us define the (i, j) commodity flow $f^{(ij)}$ as:

$$f^{ij} \triangleq \left(f_1^{ij}, f_2^{ij}, \ldots, f_b^{ij} \right)$$

where f_m^{ij} is the portion of (i, j) commodity flow flowing in arc m, and define the global flow f as:

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n} f^{(ij)}$$

In the sequel, we restrict our analysis to m.c. problems in which the performance depends solely on the global flow:

$$P(\Phi) \equiv P(f) \tag{2.3}$$

However, most of the arguments and techniques presented in the paper can be extended to the general case of $P(\Phi)$ explicitly depending upon the various types of commodities present in the system.

So far, we represented the flow configuration Φ in terms of $f^{(ij)}$, $\forall i, j$.

An equivalent representation is obtained by providing for each commodity (i, j) a set of routes $\pi_{ij}^k, k = 1, \ldots, k_{ij}$, from node i to node j, associated with some weights $\alpha_{ij}^k \left(\alpha_{ij}^k > 0, \quad \sum_{k=1}^{k_{ij}} \alpha_{ij}^k = 1 \right)$: by this we mean that commodity (i, j) is transferred from i to j along k_{ij} routes, and route π_{ij}^k carries an amount $\alpha_{ij}^k \cdot r_{ij}$ of commodity (i, j).

As a third representation, we can consider the aggregate flow Φ. It can be easily noted that Φ does not completely characterize the individual end-to-end sub-flows and routes: for instance, two different sets of routes might yield the same Φ. However, from Eq. (2.3), it turns out that such a representation is sufficient for many considerations, and is certainly more compact than the previous two. In the following we use whichever of these representations is most convenient.

It can be seen that the set of m.c. flows satisfying constraints (2.1) and (2.2) is convex. In particular, F is a convex polyhedron. The global flows corresponding to the "cor-

ners" (extreme points) of F have an interesting property: they are SINGLE-path (as opposed to multi-path) flows. In fact, by construction, they cannot be expressed as linear combinations of other flows [5]. We shall see that the flow F in any optimal flow solution is the convex combination of extreme flows, which in turn are the shortest path flows corresponding to some link costs [5].

3. Multicommodity problems in the design of S/F networks

Let us now consider a store-and-forward (S/F) communication network. In such a network, messages traveling between source and destination are "stored" in queue at any intermediate node N_k, while awaiting transmission, and are sent "forward" to N_l, the next node in the route from N_i to N_j, when channel (k, l) permits. Thus, at each node there are different queues, one for each output channel. The message flow requirements between nodes arise at random times and the messages are of random lengths; therefore the flows in the channels and the queue lengths in the nodes are random variables.

3.1. The delay model

Under appropriate assumptions, an analysis of the system can be carried out [4]. In particular, it is possible to relate the average delay T suffered by a message traveling from source to destination (the average is over time and over all pairs of nodes) to the average flows in the channels.

Definition:

"Shortest route flow" is defined as an m.c. flow in which the routes can be described by a shortest route matrix, computed for a specific assignment of lengths to the arcs.

Assumptions:

Poisson message arrivals at entry nodes,
exponential distribution of message length,
independence of arrival processes at different internal nodes,
independence assumption [4] of service times at successive nodes along the path,
independence between arrival times and service times at an intermediate node.

The result of the analysis is:

$$T = \sum_{i=1}^{b} \frac{\lambda_i}{\gamma} T_i \tag{3.1}$$

where T = total average delay per message (s/messg), b = number of arcs in the network, λ_i = the message rate on channel i (messg/s), $\gamma = \sum_{i=1}^{n}\sum_{j=1}^{n} r_{ij}$ = the total message arrival rate from external sources (messg/s), and T_i = average delay suffered by a message waiting for channel i (messg/s).

T_i is the sum of two components:

$$T_{\downarrow} i = T' + T''$$

where

$$T'_i = \frac{1}{\mu C_i - \lambda_i} = \text{transmission and queuing delay}$$

$$T''_{\downarrow} i = p_{\downarrow} i = \text{propagation delay}$$

and

$$C_i = \text{capacity of channel } i \left[\frac{\text{bits}}{\text{sec}}\right]$$

$$\frac{1}{\mu} = \text{average message length (bits/messg)}$$

We can rewrite Eq. (3.1) as follows:

$$T = \frac{1}{\gamma}\sum_{1}^{b}\left\{\frac{\frac{\lambda_i}{\mu}}{C_i - \frac{\lambda_i}{\mu}} + \left(\frac{\lambda_i}{\mu}\right)\mu p_i\right\} \tag{3.2}$$

Letting $\frac{\lambda_i}{\mu} = f_i$, Eq. (3.2) becomes:

$$T = \frac{1}{\gamma}\sum_{1}^{b}\left\{\frac{f_i}{C_i - f_i} + f_i p'_i\right\} \tag{3.3}$$

where f_i = is the average bit rate on channel i (bits/s) and $p'_i = \mu p_i$.

The average delay T is the most common performance measure for S/F networks, and the multicommodity problem consists of finding that routing, or flow pattern F, which minimizes T.

3.2. The routing assignment problem

We may now pose our main routing assignment problem:

Given:	Topology, channel capacities and a requirement matrix R
Minimize:	$T(f) = \frac{1}{\gamma}\sum_{i=1}^{b}\left(\frac{1}{C_i - f_i} + p'_i\right)f_i$
Over f	
Constraints:	(i) f is an m.c. flow (ii) $f_i \leqslant C_i$, $i = 1, \ldots, b$

The problem is in the standard multicommodity form and the additional constraints are capacity constraints. Let F_A be the set of feasible flows for this problem: $F_A = F \cap \{f | f \leqslant C\}$.

Clearly F_A is a convex set (intersection of convex sets).

In the above problem formulation, link capacities are known. The most general design also requires the selection of link capacities (given the total budget) so as to satisfy the flow requirements. This problem, known as the Route and Capacity Assignment Problem, is reported in the Appendices A–C.

3.3. The penalty function

The inspection of the routing problem motivates the following important observation:

Observation:

The performance $T(f)$ goes to infinity whenever f approaches the boundaries defined by the additional constraints (i.e., when any channel becomes saturated).

Using mathematical programming terminology, we can state that the performance $T(f)$ incorporates the additional constraints as penalty functions. From a practical point of view, such a property is very important: it guarantees the feasibility of the solution (with respect to the additional constraints) during the application of usual non-linear minimization techniques, provided a feasible starting flow is known. Techniques for finding feasible starting solutions are shown in the applications section. As a consequence of the above observation, if we assume that a feasible starting solution can be found, we can disregard the additional constraints and approach routing as an unconstrained m.c. flow problem.

4. The routing assignment

Let us consider the routing problem in Section 3.2. The performance function $T(f)$ (see Eq. (3.3)) is strictly convex (separable sum of strictly convex functions), and the feasible set F_A is a convex polyhedron. Therefore, if the problem is feasible, there is a unique stationary point, which is the global minimum. The additional constraints are included in $T(f)$ as penalties; therefore, if we can find a feasible starting flow $f^0 \in F_A$, the routing problem can be regarded as an unconstrained m.c. flow problem and solved with the FD method.

In order to find a flow $f^0 \in F_A$, several methods are available. One of them was described in [5]. Another method (applied below) consists of picking any $f \in F$, and then reducing the flows in all arcs by a scaling factor RE, until a feasible flow $f^0 = RE \cdot f \varepsilon F_A$ is obtained; f^0 satisfies a reduced requirement matrix $R_0 = RE \cdot R$. The FD method is applied using f^0 as starting flow and R_0 as starting requirement; after each FD iteration, the value of RE is increased up to a level very close to saturation. The search for a feasible flow terminates when one of the two following cases occurs: either $RE > 1$, and a feasible flow is found; or the network is saturated, $T(f)$ is minimized and $RE < 1$. In the latter case the problem is in feasible and we are finished.

The FD algorithm for the solution of the routing problem consists of two phases, Phase 1 and Phase 2. In Phase 1 a feasible flow f is found (if it exists), or the problem is declared infeasible. In Phase 2 the optimal routing is obtained. The algorithm is outlined as follows:

Phase 1:

0. With $RE_0 = 1$, let f^0 be the shortest route flow computed at $f = 0$, i.e. with metric $l_k \triangleq \left[\frac{\partial T}{\partial f_k}\right]_{f_k=0} = \frac{1}{\gamma\left(\frac{1}{C_k}+F'_k\right)}$;
   Let $n = 0$.
1. Let $\sigma_n = max_k\left(\frac{f_k^n}{C_k}\right)$.
   If $\frac{\sigma_n}{RE_n} < 1$, let $f^0 = \frac{f^n}{RE_n}$ and go to Phase 2. Otherwise, let $RE_{n+1} = \frac{RE_n(1-\varepsilon(1-\sigma_n))}{\sigma_n}$, where ε is a proper tolerance, $0 < \varepsilon < 1$.
   Let $g^{n+1} = f^n\left(\frac{RE_{n+1}}{RE_n}\right)$. Go to 2.
2. Let $f^{n+1} = FD \odot g^{n+1}$, where FD is the Flow Deviation mapping from one global flow to the next in the FD procedure.
3. If $n = 0$, go to 5.
4. If $\left|\sum_{k=1}^{b} l_k(v_k - g_k^{n+1})\right| < \theta$ and $|RE_{n+1} - RE_n| < \delta$, where θ and δ are proper positive tolerances, and v is the shortest route flow computed at g^{n+1}, stop: the problem is infeasible within tolerances θ and δ. Otherwise, go to 5.
5. Let $n = n + 1$ and go to 1.

Phase 2:

0. Let $n = 0$.
1. $f^{n+1} = FD \odot f^n$.
2. If $\left|\sum l_k(v_k - f_k^n)\right| < \theta$, where θ is a proper positive tolerance, stop: f^n is optimal within a tolerance θ. Otherwise, let $n = n + 1$ and go to 1.

The algorithm, in the form described above, provides only the optimum global flow f. If complete information about the routes taken by each commodity is required, a simple updating of routing tables at each FD iteration allows one to recover all the routes at the end of the algorithm [28].

5. ARPANET routing

We are now ready to apply the FD algorithm to the design of the ARPANET, the first computer network deployed in the US [6,7]. A detailed description of the ARPANET and its protocols is given in [8–10]. Due to the fact that new computer centers were continually added to the network, its topology was changing quite rapidly in those days, especially because each Host had to be supported by an ARPANET router, IMP or TIP [11–14]. In this paper we refer to the 21-node, 26-link topology in Fig. 3.

For simplicity (and with no impact on convergence) we assume that the traffic requirement between all pairs of nodes is uniform. Namely, traffic is $r = 1.187$ (Kbits/s), uniform for all node pairs. We want to find the routes that minimize the delay expression reported in Eq. (3.3) and plotted Fig. 2. First, we note that, for the 21-node ARPANET with a uniform requirement, the "large and balanced net" condition holds. In other words, the condition for performing individual "flow deviations" (one path and thus one single flow per source/destination pair) is satisfied [19].

It makes sense to apply both optimal (multipath) and single-flow (and single-path) FD algorithms and compare the results. These are: (a) for the optimal FD algorithm $T_{\min} = 0.2406$, obtained after 80 iterations, with accuracy = 10^{-4} T; for the single-path (non-bifurcated) FD algorithm $T_{\min} = 0.2438$ s, obtained after 12 shortest-path computations. Both algorithms were programmed in Fortran and run on an IBM 360/91; the execution time was 30 s for the optimal algorithm and 4 s for the single-path one. The error of the suboptimal single-path solution is less than 2%. This fact shows the power of the large and balanced nets property and the practical value of the single flow solution.

Fig. 2 illustrates the application of the non-bifurcated (single-path) algorithm. Recall that RE is the traffic level normalized to $r = 1.187$ Kbits/s. The traffic is first routed

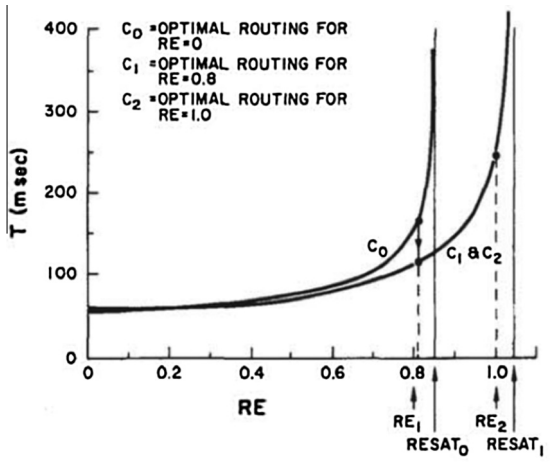


Fig. 2. Average delay T versus normalized traffic RE, using various routing schemes.

along the shortest routes computed for $RE_0 = 0$; curve C_0 plots the delay T versus RE, using such a routing scheme (which we refer to as RS_0). With RS_0, the saturation level for the traffic is $RESAT_0 = .85 < 1$; $RE = 1$ is infeasible, and therefore we are still in Phase 1. Let f^1 be the flow obtained by routing traffic level $RE_1 = .95\ RESAT_0 \cong .8$, according to RS_0 and apply to f^1 the FD algorithm; a new routing scheme RS_1 is obtained, which improves $T(RE_1)$. Curve C_1, corresponding to RS_1, saturates at $RESAT_1 = 1.05 > 1$; $RE = 1$ is feasible and Phase 2 is initiated, with $RE_2 = 1$. At the end of Phase 2, the sub-optimal, non-bifurcated routing scheme RS_2 is found; curve C_2 corresponding to RS_2 practically coincides with curve C_1, in Fig. 2, as the scale of T is not detailed enough to show differences in values. Notice that, as expected, the routing RS_0 gives the best results at low traffic levels; in fact, RS_0 is almost optimal up to $RE = 0.5$.

6. Lessons learned and follow-up work in the ARPANET scenario

The FD algorithm can be applied to any unconstrained m.c. flow problem and leads to a global optimum when some reasonable assumptions on $P(f)$ convexity are satisfied. It can also be applied to constrained flow problems when the constraints can be included in the objective function as penalties. For this reason, the FD algorithm has been an efficient tool for the design of packet networks and in particular the maintenance of the ARPANET in the 1970s. The computation time per iteration required by FD was comparable to that of the prevailing heuristic techniques at that time [2].

The ARPANET generated a lot of interest in networks in general and in routing and flow assignment in particular. Within this framework, the FD multicommodity flow approach inspired work in many aspects of ARPANET protocol design. The first challenge was the design of an operational routing algorithm for the ARPANET based on distributed computation. The existing Bellman-Ford Algorithm used in the ARPANET at the time was providing only one path, the shortest path [2]. An "alternating path" protocol based on a Bellman-Ford extension had failed because of oscillations triggered by packet queue dynamics [19]. The FD algorithm promised an improved solution based on the systematic selection of improving multiple paths. However, the FD method was designed for quasi-static networks so it could not be applied directly to the ARPANET where traffic is very dynamic. Moreover, the FD algorithm is centralized while the ARPANET routing protocol must be fully distributed for resilience to failures and to support scalability. Let us recall that, initially, the FD algorithm was used mainly in ARPANET topology management, i.e. to verify that new link additions would provide the best growth in throughput capacity. The institution charged by DARPA with the topology upgrade and management task

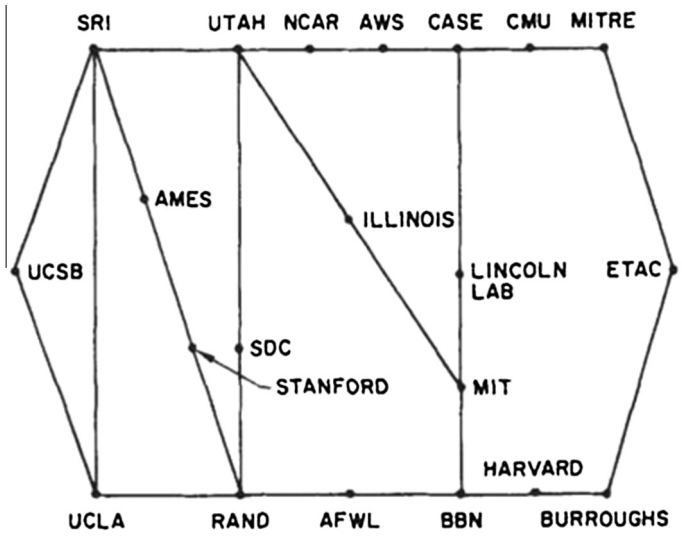


Fig. 3. 21-node ARPA topology.

in the early 1970s was Network Analysis Corporation, a small study house in New York, the precursor of future Bay Area startups that mushroomed during the Internet explosion 30 years later. Gerla worked at Network Analysis Corporation in that period and cranked out efficient designs using the FD algorithm for flow assignment as well as joint flow and capacity assignment.

The challenge of employing multicommodity flow theory, and more precisely embedding the "incremental improving flows" technique into the ARPANET operational protocols was taken on by Gallager [29]. Gallager in 1977 designed a very elegant method that monitors traffic rates and delays in real time and optimizes routing tables accordingly [29]. His method was implemented and validated in a DARPA experimental testbed. Shortly after, Gallager and Golestani extended the basic routing scheme by introducing real-time optimal rate control. Namely they used FD techniques to optimize a convex objective function consisting of the sum of weighted throughput benefits minus the congestion penalties [30]. However, in spite of the conceptual elegance of the approach, the distributed Gallager algorithms did not find significant adoption in the operational routing protocols due to non-stationarity of the ARPANET flows, relative complexity of the scheme, and scarce computational resources of routers in those days. It will take several more years before multicommodity flow techniques will be used for on-line, operational traffic management.

7. Flow Deviation beyond the ARPANET

After the ARPANET applications in the 1970s, the interest in the FD algorithm persisted in the networking research community and inspired similar methods for a broad range of scenarios involving not just packet switched networks. In fact, as the networking technology evolved, new "flow type" problems faced designers, who found it convenient to formulate them as m.c. flow problems, resorting to FD type techniques to solve them. It is remarkable that interest in the FD method persisted in spite of the fact that since the FD algorithm was first published, many other more sophisticated and computationally more efficient techniques have been proposed based on powerful linear and mathematical programming modules [5]. We suspect that the simplicity and the intuitive appeal of the seminal FD concept have inspired generations of researchers to use it instead of (or in addition to) more powerful methods. It must be said that a simple algorithm like the FD algorithm allows the designer to remain intuitively coupled with the steps of the optimization, and to interact with the design by changing some of the parameters. The linear/integer programming packages often used for these optimizations, apart from being complex to set up and time consuming to run, generally offer little insight in the design process.

Whatever reason motivated network designers to use the FD scheme, the fact is that more than 500 papers have cited the original FD paper. It is interesting to peruse these references to appreciate the breadth in scope and scenarios of these later FD applications. Many of these works apply FD to find min-delay routing solutions in customized packet networks with characteristics that differ from the original ARPANET packet network design. Some other papers apply the FD method to flow problems where the flows are not packet flows. In this section we focus on applications other than packet switching. We will discover that most of these case studies demonstrate the key value of the FD method, namely its simplicity, generality and flexibility of use. For the sake of brevity, we will limit ourselves to a few representative cases ranging from reliable topology/routing design in the face of failure/loss to wavelength channel routing in WDM optical networks Hybrid Wireless-Optical Broadband Access Network (WOBAN) design and power savings for an Internet service provider (ISP). The applications in this section are all about data networking, albeit not conventional packet-switched networking. In the next section we will present another FD application, this time to a network that has nothing to do with data – namely a vehicular network.

The first paper we reviewed was published in 1998 [15]. It is about reliable design of networks subject to failures/loss. As the network complexity increases, the notion of reliability to failures and losses gains greater importance. The design of large transmission networks under reliability constraints becomes a complex non-linear optimization problem, typically requiring a large amount of computation time to produce an optimal solution. The authors simplify the problem using heuristics. They reduce it to a classical network synthesis problem and employ well-known multicommodity flows (MF) methods for which there is a large body of work, both theoretical and numerical. In particular, they use the flow deviation method after transforming their capacitated MF model into an uncapacitated version using penalty functions. They then generalize their approach proposing an FD-based methodology for network synthesis with reliability constraints.

Two years later in 2000, the authors of paper [16], use the FD method to route different wavelengths over a Wavelength Division Multiplexing (WDM) network. In WDM-based packet networks, routers are connected to each other using wavelengths (lightpaths) to form a logical network topology. This logical topology may be reconfigured by rearranging the lightpaths that connect the routers and by load balancing them so as to minimize the maximum link load. Finding the topology that minimizes the maximum link load would require the joint solution of the lightpath connectivity problem and the traffic routing problem, leading to an NP-complete problem. To simplify matters, the two problems are solved separately. Flow Deviation has been used to find the optimal routing that minimizes the maximum link load for a given topology configuration.

In 2009, paper [17] proposes an access network that uses combined radio and optical links to optimize cost/performance tradeoffs. Wireless nodes collect traffic from end users and carry them to the optical part of the WOBAN using multiple radio hops. Since each node has a single radio interface, the multiple radio hops accumulate delays and limit capacity. Thus, the radio mesh network is the critical bottleneck. To overcome this limitation, a capacity and delay aware routing scheme is proposed. The FD

method is used to build CaDAR, a capacity and delay aware routing scheme, achieving much higher loads and greatly improving the performance of conventional routing schemes, yet within acceptable delays.

In 2011 paper [18] addresses the problems of energy consumption and greenhouse effects. The authors propose to use information and communication technologies (ICT) to reduce the energy budget through the optimization of energy generation, transportation, and consumption. A good starting point, in this context, is to reduce the overall power consumption of an Internet service provider (ISP) backbone network, considering it as a single, large, and distributed system. Their goal is to find the minimum set of devices that must be used in order to control the whole network and meet the actual traffic demands. The latter is achieved by managing the over-provisioned capacity needed to satisfy quality-of-service (QoS) constraints during peak loads. To achieve this, they propose to selectively turn on and off the network devices to save energy while guaranteeing the requirements. This optimization can be precisely defined by an integer linear programming (ILP) formulation at different complexity levels. The authors defeat the complexity by aggregating variables and computing the associated flows. In this formulation, where only real variables are involved, the FD algorithm can easily provide an optimal solution to the routing of the aggregated flows.

8. Routing of vehicular flows using FD principles

The number of vehicles in most urban areas has outgrown road capacity, leading to severe traffic congestion during rush hour. Urban planners are looking for solutions to reduce traffic congestion. Real-time traffic information obtained from vehicles can help reduce congestion by exploiting the DSRC (Dedicated short-range communications) wireless architecture recently established for safety and traffic control purposes [34]. To this end researchers at UCLA have proposed NAVOPT, a vehicular routing navigator as well as a strategy implemented in the navigation server and assisted by the on-board navigators [31]. In current modern navigation systems, the on-board navigator equipped with area map and GPS monitor, periodically reports its own position to the server via wireless connection (Wi-Fi or 3G); in turn, the server returns to the navigator the minimum cost path (i.e. path with shortest travel time) under the current traffic conditions. In NAVOPT, the server uses the FD algorithm to compute optimal vehicle routes that load balance vehicle traffic over the network. Below we review the analytic models and assumptions of NAVOPT and present a synopsis of simulation results generated by the SUMO simulator.

8.1. NAVOPT architecture

In NAVOPT, we assume vehicles are equipped with a smart navigation system that supports wireless communications and features geographical map information. The navigation system has a Global Positioning System (GPS) and WiFi (i.e. 802.11a/b/g) or dedicated short-range communications (DSRC) (i.e. 802.11p and WAVE) radio interfaces [34]. The access points are called road side units (RSUs). They constitute a wireless mesh network (WMN) and provide Internet access to moving vehicles. The vehicles organize V-GRID, namely a mobile mesh network among the vehicles. In this NAVOPT architecture, the vehicles can communicate with each other as well as with servers located in the Internet through the RSUs.

NAVOPT is implemented in a centralized way. The city is subdivided into partitions. Each geographical partition has a designated navigation server that provides optimal travel routes to drivers. Large-scale vehicular networks such as nation-wide networks can be segmented into many small-scale networks (e.g. city networks) in a hierarchical system that maintains scalability.

Vehicles gather traffic information such as vehicle density, speed, and accident locations using various on-board sensors like cameras and GPS. In a typical NAVOPT scenario, some vehicles may not report their traffic information due to the fact that they are selfish or do not have wireless connections. However, NAVOPT can improve vehicular routing even with partially observed traffic information and with control on only some of the vehicles. A typical vehicle is equipped with a communication module for sensed data distribution using DSRC or other radio transceivers, a sensing module, a processing module and a database module for traffic information management. Vehicles report periodically their sensed information, e.g. position, average speed, travel direction, etc., to a navigation server. Based on such information, the navigation server can estimate travel time as a function of average number of vehicles on the road and delay to travel the road segment. The delay can also be derived from the density of the road based on experimentally validated numerical models. However, exceptional cases like accidents can alter the relationship between them. In the sequel, we limit ourselves to traffic distributions that correspond to normal traffic conditions.

8.2. Multicommodity flow problem in vehicular routing

This section gives a design overview of NAVOPT, navigator assisted optimal vehicular routing. We can model the stop-and-go traffic at each intersection as the packet traffic in a communication model. A road intersection is the functional node of the vehicular grid, just as a router that selects a route and forwards packets is the node of the communications network. In this context, we can transfer to vehicles the general queuing model of the communication network. The use of multicommodity flow models for vehicular traffic management had been proposed before, in fact, even before the ARPANET routing formulation. One of the earliest vehicular flow models was described by Dafermos and Sparrow [26]. Optimization of vehicular flows however was hindered by the difficulty of collecting realistic traffic volumes (i.e. source destination requirements). Also, once optimal routes were computed, it was difficult to communicate such routes to drivers. Today, the on-board navigator serves both functions – as a sensor of traffic and as a route enforcer [31].

Recall that the general multicommodity flow problem is finding best routes for multiple flows while minimizing total cost of routing (e.g., end-to-end delay) [19]. This vehicular routing optimization problem can be formulated as follows:

$$\min \sum_{s,f} P(\Phi), \quad f \in F,\ s \in G$$

where $P(\Phi)$ is a performance measure of the multi-commodity flow (e.g. delay or cost) for a given flow configuration Φ. F and G are sets of flows (of vehicles) and nodes (i.e. intersections) in a given network topology (i.e. urban grid). Requirement flow, r^f is an incoming flow from the outside world. The flow is constrained by the following network flow conservation rule and must be positive.

$$\sum_{j \in G(j \neq i)} x_{ji}^f + r^f_{(s(f)ord(f)=i)} \leqslant \sum_{j \in G(j \neq i)} x_{ji}^f \quad \forall f, i, j \tag{8.1}$$

where the r^f is positive if its source is node i and negative if its destination is node i.

$$x_{ji}^f > 0 \quad \forall f, i, j \tag{8.2}$$

When a multicommodity global flow f satisfies the above constraints Eqs. (8.1) and (8.2), the set of feasible flows F spans a convex polyhedron. A local extreme point in F corresponds to a set of non-bifurcated (i.e. not split on multiple path) flows [3].

From now on, we assume that the vehicular routing performance function, $P(\Phi)$, can be expressed by a delay penalty where the flow configuration Φ is the global grid flow assignment. The average travel time T is the end-to-end delay from source to destination:

$$T = \sum_{e=1}^{b} \frac{\lambda_e}{\gamma} T_e \tag{8.3}$$

where T is total average delay (h/vehicle) of the aggregate flow, b is number of road segments, λ is arrival rate of a flow at each road segment e and γ is the traffic requirement (total vehicle arrival rate to the system) from the external world. A requirement flow r^f from node i to j) can be written as r_{ij}, where r_{ij} is a $n \times n$ matrix at the given network that consists of n nodes (i.e. intersections) and b roads (i.e. total road segments in network). Then, total aggregate incoming flow is:

$$\gamma \left[\frac{vehicles}{hour} \right] = \sum_{f}^{F} r^f = \sum_{i=1}^{n} \sum_{j=1}^{n} r_{ij} \tag{8.4}$$

T_e, the average delay for road e, is composed of queuing (T_q) and propagation delay (T_p), namely the time to wait for a traffic signal at an intersection and the time to move across the intersection, respectively. It is expressed by:

$$T_q = \frac{1}{C_e - \lambda_e} \tag{8.5}$$

where λ_e is the arrival rate (e.g. vehicles/h) at each road segment and C_e is the road capacity (e.g. vehicles/h). As shown in 8.5, T_q is the main input parameter for the vehicular routing algorithm. It is eventually determined by vehicle density of each road e.

$$T_p = \frac{Road\ Length}{Max\ Speed} \tag{8.6}$$

With the average delay (T_e), the equation Eq. (8.3) may be rephrased as follows:

$$T = \sum_{e=1}^{b} \frac{\lambda_e}{\gamma} T_e = \frac{1}{\gamma} \sum_{e=1}^{b} \frac{\lambda_e}{C_e - \lambda_e} + \lambda_e T_p \tag{8.7}$$

Fig. 4 depicts an example of vehicle formation to help describe each parameter mentioned above. The road is partitioned into 400 m segments with a single lane for each direction. A maximum speed of 60 km/h is allowed as is typical in urban roads according to common speed policies. Thus, travel delay (T_p) per each road segment for example can be 400 m/(60 km/h) = 0.0067 (h). An approximate maximum density in each lane of the road e is around 400 m/4 m = 100 vehicles when we assume the vehicle length is 4 m. Thus vehicle density in the road is 250 (vehicles/km). Density can vary with vehicle type (e.g. bus or trailer) and safety gap between vehicles. Based on this approximated density, road capacity C_e is derived as 100 vehicles/0.0067 h = 14,925 (vehicles/h) when we ignore micro-mobility issues such as deceleration in intersections. Our model is useful for investigating the efficacy of the FD algorithm for vehicular routing. However, the actual traffic is quite different from the traffic estimated by this numerical approach because of the many extra factors that impact our analysis.

NAVOPT's objective is to find the route assignment that minimizes delay expression (8.7). As discussed earlier, the capacity constraint is incorporated in the objective as a penalty function. In NAVOPT, this unconstrained multicommodity flow problem is solved by the FD method. The Flow Deviation method is an iterative method. It evaluates at each iteration the direction of the steepest descent. NAVPOT processes repeated Flow Deviations to find the optimal traffic distribution characterized by the stationary condition $T(fn) - T(fn+1) < \varepsilon$.

8.3. Evaluation

We have first established a numerical model of NAVOPT using the Flow Deviation algorithm described in the previous section and have run simulations with the SUMO traffic simulator. Fig. 5 shows a road topology to evaluate NAVOPT performance, namely a Manhattan grid where each road segment is 400 m and maximum allowed speed is 60 km/h. Vehicle length is 4 m. Acceleration and deceleration are 4 and 4.5 respectively. Driver noncompliance (σ) is 0.5. Here we assume a flow from a source denoted by S to a destination D as can be seen in Fig. 5. A set of routes (i.e. from F1 to F5) from the source to the destination consists of a shortest path and four alternate paths. Therefore, the routes have different path lengths. For example, the shortest path F1 has 9 road segments and F2 and F3 have 11 road segments.

8.4. Numerical analysis

Using NAVOPT, performance was evaluated in terms of average delay, average speed and average density under

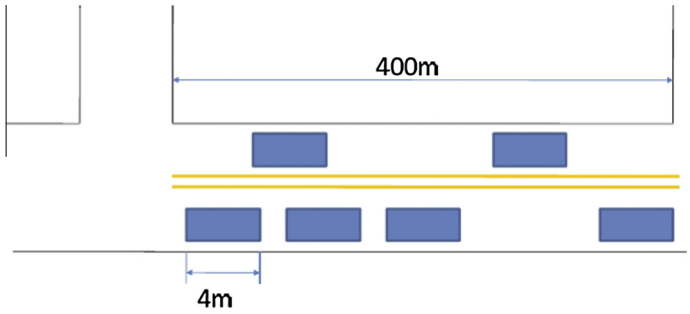


Fig. 4. Model of vehicle formation on the road.

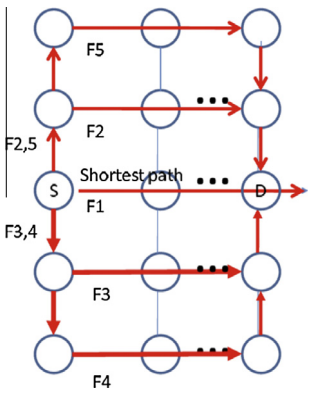


Fig. 5. Manhattan grid topology.

varying traffic requirements. Fig. 6a plots performance comparison between the NAVOPT multipath solution and the shortest path solution provided by conventional navigators. As can be seen in Fig. 6a, the delay of the shortest path solution increases rapidly above single segment capacity = 14,800 (vehicles/h), while the average delay of each route in NAVOPT remains constant well beyond the saturation point because the flow is deviated to different routes. In Fig. 6c, average density exhibits a behavior similar to average delay. According to a fluid traffic model (FTM), the speed of a vehicle can be derived by the current vehicle density over the road [20] as follows:

$$v(t+\Delta t) = \max\left[v_{\min}, v_{\max\left(1-\frac{\rho(t)}{\rho_{\max}}\right)}\right]$$

where the $v_{\min}$ and $v_{\max}$ are allowed minimum and maximum speed, and $\rho(t)$ is the current vehicle density and $\rho_{\max}$ is the maximum vehicle density of the road.

Fig. 6b depicts the average speed of the road segment. Average speed in NAVOPT is close to the maximum legal speed. In contrast, in the shortest path solution speed decreases rapidly as density increases. Once the density reaches the maximum, i.e. 250 vehicles/km, the road locks up and becomes a parking lot as the vehicle speed reduces to zero.

8.5. SUMO simulation results

We validated the numerical NAVOPT results with a NAVOP implementation integrated with the SUMO-0.12.0 simulator [33]. SUMO supports a micro-mobility model for each vehicle as well as the end-to-end flow model. We generated a Manhattan grid map with multiple flows from the same source and destination. Initially, all the flows use the shortest path. Then, they are rerouted to multiple paths as provided by the background NAVOPT numerical optimization. Performance is evaluated (in terms of average delay, speed and density) for increasing inflow rate values, both under shortest path and under multiple paths set by NAVOPT.

Average delay for each road segment is shown in Fig. 7a. Average delay in a road increases drastically when the incoming flow rate approaches 14,800 (vehicles/h) in the case of the shortest-path scenario. In contrast, NAVOPT shows almost constant delay even if the incoming flow rate increases. It is worth noting that the delay does not go to

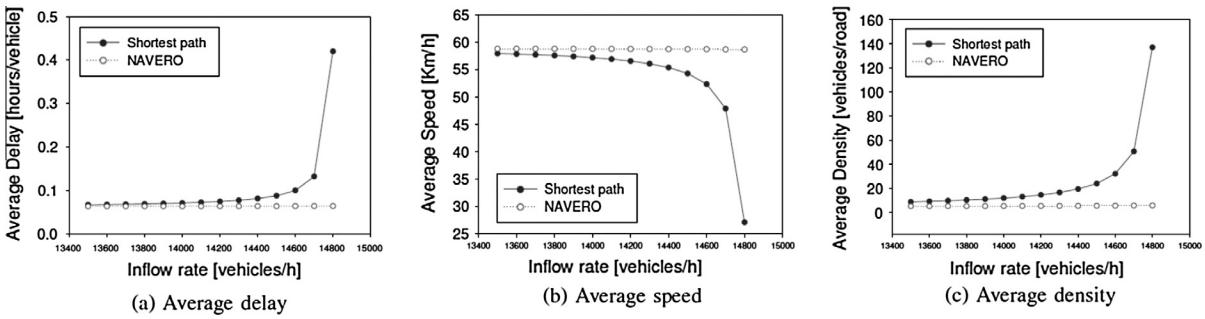


Fig. 6. Performance results from numerical model.

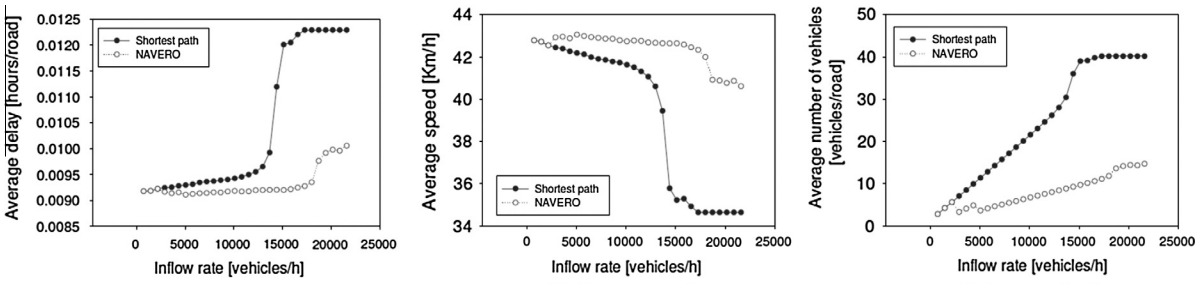


Fig. 7. Performance results from SUMO simulation.

infinity since the simulator automatically reduces the input traffic as congestion sets in. The same result can be seen in Fig. 7b. The road density becomes saturated as the incoming flow rate is increased. This points out another difference between the numerical model and simulator, namely the maximum vehicle density. A maximum of 250-vehicle capacity per each road segment was conjectured in our numerical model; the simulator peaks at 40 vehicles. Also, vehicle density in NAVOPT increases very slowly as the incoming flow rate grows. As for average speed in each road, vehicles in NAVOPT maintain an average speed near 42 km/h, while vehicles in the shortest path solution slow down to 34 km/h.

In all, the FD algorithm has proven efficient in rerouting vehicular traffic to alternate routes. This is supported by analytic models as well as by realistic, state-of-the-art simulation results. NAVOPT can improve speed and throughput by about 25% compared to shortest path routing (the "selfish" routing implemented by individual drivers following the shortest path). Also, NAVOPT reduces total travel time by 40%. NAVOPT can improve throughput even with partial penetration based on numerical analysis. The agreement between analytic results and simulation results and the robustness to low penetration demonstrates that FD is a valid vehicular routing method. In fact, FD works much better with vehicle flows than with ARPANET packet flows, since vehicle dynamics are much slower than FD background computations. The latter, in turn, are much slower than packet dynamics.

9. A future opportunity for the FD algorithm: SDN tunnel routing

The FD principle of optimizing flows by flow increments (i.e. "flow deviations") clearly works well with convex objective functions. It also works well as an operational, on-line routing protocol if the Flow Deviations can be computed and implemented in the network much faster than the changes in the underlying flows. In traditional datagram packet networks, packet sessions can be very bursty and flows change within milliseconds, thus preventing the use of the FD algorithm for packet-by-packet operations. That is why the distributed routing scheme based on the FD method developed by Gallager et al. in the mid 1970s [29] did not meet much practical success at the time. Recently, however, a new network technology has emerged – the Software Defined Network. SDN is based on the open flow model where the unit is not the packet, like in traditional "datagram" nets, but the flow of packets between a source and a destination. More generally, for scalability, the flow is defined as the aggregate of individual flow sessions, all with similar properties and similar performance requirements that are carried in the same source-to-destination tunnel. Individual sessions are managed by TCP. In addition, edge routers can perform rate control on each tunnel as extra protection from congestion and for QoS performance and fairness guarantees. SDN solutions have become popular in data center networks (like Google, Amazon, and Facebook). They interconnect massive clusters of computers scattered all over the world and carry gigabit/s streams on dynamically reconfigured tunnels.

Given the high volume of traffic in data centers, the available communication channels (typically multiple 40 Gbps channels laid out on different geographic paths) must be efficiently used. In SDNs the routing problem is a problem of routing not individual packets as in the ARPANET, but entire tunnels. Flows in tunnels are fairly stable in time, with minute (as opposed to ms) dynamics. Generally, multiple tunnels are maintained between each source–destination pair. Minor changes in traffic demands are handled by redistributing the S–D traffic over the tunnels. Major changes require the establishment of new tunnels. Given the centralized nature of a data center system, the route optimization is done from a central server. Flow stability and central control favor the use of multicommodity flow optimization schemes. In fact, the authors of [32] propose SWAN, a software-driven WAN, "*...that boosts the utilization of inter-datacenter networks by centrally controlling when and how much traffic each service sends and frequently reconfiguring the network's data plane to match current traffic demand.*" However, the authors lament that the conventional LP methods are too cumbersome and time-consuming for the data center application and thus introduce heuristics to solve the problem. The presence of multiple constraints, namely channel capacity and node flow table entry limits (recall that each tunnel requires one separate flow entry), make the problem rather complex. In SWAN the optimization is carried out in two steps: selection of a new set of tunnels (that minimize path lengths and comply with flow table constraints), and optimal routing of the traffic on these tunnels, so as to maximize the throughput while maintaining fairness.

These two steps are reiterated until a feasible solution that can accommodate all the traffic and comply with capacity and flow table constraints is found.

This problem can also be attacked with the an FD-like algorithm by incorporating the constraints as penalties in the objective function. With the FD approach, at each iteration a new "shortest path" tunnel (the equivalent of shortest path in the conventional FD algorithm) is found, and traffic is "deviated" from current tunnels to the new tunnel. In the process, one or more old tunnels may run "dry" and are eliminated, freeing up flow table entries. Fairness across different source/destination pairs is automatically supported by the FD algorithm as a byproduct of the incremental flow optimization. In general, an individual session is transported entirely within a single tunnel. However, in the case of a very large single session flow that exceeds the capacity available on a single path, the FD method allows the system to split the flow over multiple paths. Distributing a TCP flow on multiple paths is made possible today by a new IETF TCP protocol version called MP TCP (http://tools.ietf.org/html/rfc6824).

10. Conclusions

The FD algorithm was proposed in the early ARPANET days to assist in the design of efficient packet-switched topologies. The main attraction of the FD algorithm was its simplicity and the ability of intuitive interpretation of the various optimization steps, which helped the network researcher develop design techniques closely matched with the physical properties of the system. This simplicity and yet scientific accuracy made the FD algorithm popular in many different domains. The incremental flow concept applies best to problems with quasi-stationary flows. The early ARPANET traffic was very bursty, so the FD method could not be used for direct traffic control on a dynamic basis. However, rush hour traffic in a vehicular grid can be well handled in real time by FD centralized optimization because vehicular motion dynamics are much slower than message exchange and FD computation dynamics. Another area where FD can be successfully applied for real-time flow routing is that of software-defined networks. The FD method may offer new insight to designers in the dynamic reconfiguration of the SDN tunnels.

Appendix A. The Route and Capacity Assignment Problem

Another interesting problem in S/F networks is the route and capacity assignment problem. This problem was the original problem that motivated the FD study and paper. The problem however is quite specialized to S/F networks and has not been followed by many other researchers. For the sake of historical accuracy it is reported here in the Appendices A–C.

The route and capacity assignment problem is formulated below. Assume that we have a given network topology in which the channel capacities have to be assigned. A cost is associated with the values of the capacities, and the total cost of the network is given. In addition, the flow routes must he determined. The problem statement is:

Problem B: "Routing and capacities assignment, general cost-capacity function"

Given:	Topology, requirement matrix R, number of dollars available D
Minimize:	$T(C,f)=\frac{1}{\gamma}\sum_{i=1}^{b}\left(\frac{1}{C_i-f_i}\right)+P'_i f_i$
over C, f	
Constraints:	(i) f is an m.c. flow (ii) $f_i \leqslant C_i$, $i=1,\ldots,b$ (iii) $\sum_{i=1}^{b} d_i(C_i) \leqslant D$ where $C=(C_i,C_2,\ldots,C_b)$ $d_i(C_i)$ = arbitrary cost-capacity function for arc i

The minimization can be carried out first on C, keeping f fixed, and then on f.

If the cost-capacity functions are linear (i.e., $d_1(C_1)=d_i(C_i)$, then the minimization over C can easily be performed by the method of Lagrange multipliers and we get the following optimum capacities as functions of the flows [4]:

$$C_i = f_i + \frac{D_e}{d_i}\frac{\sqrt{f_i d_i}}{\sum_{j=1}^{b}\sqrt{f_j d_j}} \tag{11.1}$$

where

$$D_e = D - \sum_{i=1}^{b} f_i d_i$$

By introducing Eq. (2.4) into the expression of $T(C,f)$ we have:

$$T(C,f)=T(f)=\frac{\left(\sum_{i=1}^{b}\sqrt{f_i d_i}\right)^2}{\gamma D_e}+\frac{1}{\gamma}\sum_{i=1}^{b} f_i P'_i \tag{11.2}$$

Since

$$D \geqslant \sum_{i=1}^{b} d_i C_i \quad \text{for (iii)}$$

and

$$\sum_{i=1}^{b} d_i C_i \geqslant \sum_{i=1}^{b} d_i f_i \quad \text{for (ii)}$$

then

$$D \geqslant \sum_{i=1}^{b} d_i f_i$$

and

$$D_e = D - \sum_{i=1}^{b} d_i f_i \geqslant 0 \quad \text{(iv)}$$

It is easy to see from Eq. (11.1) that (iv) also implies (ii) and (iii); hence both (ii) and (iii) can be replaced by (iv).

By introducing Eq. (11.2) into Problem B′ and using result (iv), we obtain:

Problem B: "Routing and capacities assignment, linear cost-capacity function"

Given: Topology, requirement matrix R, number of dollars D

Minimize: $T(f) = \frac{\left(\sum_{i=1}^{b}\sqrt{f_i d_i}\right)^2}{\gamma D_e} + \frac{1}{\gamma}\sum f_i P_i'$

over f

Constraints: (i) f m.c. flow
(ii) $D_e \geqslant 0$

Again the problem is reduced to an optimal flow problem of the standard multicommodity form. The additional constraint is now a cost constraint. Let F_B be the set of feasible flows for Problem B:

$$F_B = F \cap \left\{ f \middle| D - \sum_{i=1}^{b} d_i f_i \geqslant 0 \right\}$$

Clearly F_B is convex.

Appendix B. Classes of m.c. flow problems

In order to place the Flow Deviation (FD) method in the proper perspective in relation to the existing methods, it is convenient to classify the various m.c. flow problems into categories. For each category, the solution techniques available in the literature are reviewed and the contribution of the FD method is discussed.

B.1. Unconstrained m.c. flow problems

B.1.1. Linear performance

The linear min-cost flow problem with no constraints on capacity has the well-known shortest path solution (where the arc length is equivalent to the linear cost of the arc) [5,22]. Very efficient techniques are available for the evaluation of all shortest routes on a graph and for the routing of the commodities along such routes [3,23]. Therefore it appears convenient to reduce complicated flow problems (i.e., non-linear, or constrained) to the linear, unconstrained form, which can be solved efficiently.

B.1.2. Non-linear performance

The most natural thing to do is to linearize the problem. Problems which are separable and convex can be linearized by approximating the convex functions with piecewise linear functions and by introducing one supplementary variable and one constraint equation for each linearized segment [21,24,25]. This method has two serious drawbacks: first, it can be applied only to separable and convex problems; second, the number of variables and constraints becomes prohibitively large for large networks.

Another method, which applies to differentiable problems, consists of approximating the performance function with the tangent hyperplane, which is expressed in terms of the partial derivatives. The min-cost solution of the linearized problem is the shortest route flow. As will be shown later, such a shortest route flow represents the direction of the steepest descent Flow Deviation.

A separable m.c. flow problem has the form:

$$P(f) = \sum_{i=1}^{b} P_i(f_i)$$

The above idea is the essence of the FD method, which consists of repeated evaluations of steepest descent directions and of one variable minimization along such directions; the method (described in Section 4) is conceptually very similar to the gradient method applied to nonlinear minimization problems. If the problem is differentiable, the FD method is clearly superior to the supplementary variables method mentioned before: it does not add new variables and constraints, and can be applied to non-convex, non-separable cases.

In fact, the idea of using shortest routes (computed with partial derivatives) for the solution of nonlinear problems is not new: using such techniques, Dafermos and Sparrow [26] solved various traffic problems, formulated as unconstrained, convex m.c. flow problems, and Yaged [27] solved a min-cost capacity assignment for a communications network, which was formulated as an unconstrained, concave m.c. flow problem.

Dafermos stated the conditions for the optimality of the solution and proposed an algorithm for finding the optimal routing in the convex case; the algorithm, however, is impractical for large nets, as it requires the bookkeeping of all paths for all commodities [26]. Yaged's results, on the other hand, are very restricted: they apply only to a separable, concave problem [27].

Here, we attempt a more general, systematic investigation of the method; we introduce the main results in a more straightforward way and in a simpler formulation than in [26]. We indicate an algorithm which is applicable to non-separable problems and which has been efficiently applied to large nets.

B.2. Constrained m.c. flow problems

B.2.1. Linear performance, linear constraints

The classic and most efficient approach is the Dantzig–Wolfe decomposition [22,28], which reduces the solution of the main problem to the repeated solution of a master problem and a subproblem. The master is a linear program containing the additional constraints, and the subproblem, which generates new columns to introduce into the master, is an unconstrained linear min-cost flow problem.

B.2.2. Nonlinear performance, nonlinear constraints

The general theory of nonlinear problems with nonlinear constraints is very hard. The special case of convex performance and concave non-negativity constraints, however, can be attacked efficiently with the Dantzig–Wolfe decomposition for convex programs [24]; the master problem is a linear program, and the column generating subproblem is an unconstrained convex min-cost flow problem. Here is another important area of application for the FD method.

We showed that the two design problems considered in the paper can be regarded as unconstrained m.c. flow problems; therefore, in the sequel, unless otherwise specified, we disregard constraints and consider only unconstrained problems.

Appendix C. Optimality conditions

Let us check if $T(f)$ satisfies the conditions for the convergence (see Section 5 in [19]). The first and second partial derivatives are:

$$\frac{\partial T}{\partial f_i} = \frac{1}{\gamma}\left[\frac{C_i}{(C_i - f_i)^2} + P_i'\right] \tag{13.1}$$

$$\frac{\partial^2 T}{\partial f_i \partial f_j} = \begin{cases} 0 & \text{for } i \neq j \\ \frac{1}{\gamma}\frac{2C-i}{(C_i - f_i)^2} & \text{for } i = j \end{cases} \tag{13.2}$$

From Eq. (2.3) in [19], the optimal solution f^*, if it exists (i.e., if the problem is feasible), satisfies the capacity constraints as strict inequalities $(f_i^* < C_i \forall i)$. Therefore, we can find an

$\varepsilon > 0$ *s.t.* :

$$f^* \varepsilon\ F_A' \triangleq F \cap \{f | f_i C_i - \varepsilon\} \tag{13.3}$$

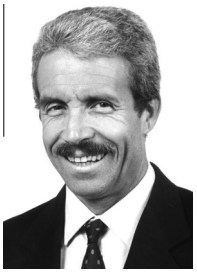
The application of the FD method can be restricted to F_A-$'CF_A$. For $f \varepsilon F'$ the sufficient conditions on the first two derivatives of $P(f)$ are satisfied; therefore the FD algorithm converges to the global minimum

References

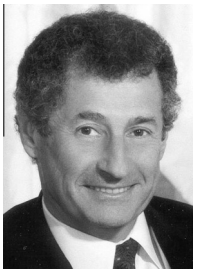
[1] L. Kleinrock, Analytic and simulation methods in computer network design, in: Proceedings of the Spring Joint Computer Conference, Atlantic City, New Jersey, May 5–7, 1970, pp. 569–579.
[2] H. Frank, I.T. Frisch, W. Chou, Topological considerations in the design of the ARPA computer network, in: Proceedings of the Spring Joint Computer Conference, Atlantic City, New Jersey, May 5–7, 1970, pp. 581–587.
[3] T.C. Hu, Integer programming and network flows, in: DTIC Document, 1969.
[4] L. Kleinrock, Communication Nets: Stochastic Message Flow and Delay, McGraw-Hill Book Company, 1964 (Out of Print) Reprinted by Dover Publications, 1972 and in 2007.
[5] D.G. Cantor, M. Gerla, The Optimal Routing of Messages in a Computer Network Via Mathematical Programming, p. 167.
[6] L.G. Roberts, Multiple computer networks and intercomputer communication, in: Proceedings of the First ACM Symposium on Operating System Principles, 1967, pp. 3.1–3.6.
[7] L.G. Roberts, B.D. Wessler, Computer network development to achieve resource sharing, in: Proceedings of the Spring Joint Computer Conference, Atlantic City, New Jersey, May 5–7, 1970, pp. 543–549.
[8] F.E. Heart, R.E. Kahn, S.M. Ornstein, W.R. Crowther, D.C. Walden, The interface message processor for the ARPA computer network, in: Proceedings of the Spring Joint Computer Conference, Atlantic City, New Jersey, May 5–7, 1970, pp. 551–567.
[9] S. Carr, S. Crocker, V. Cerf, HOST-to-HOST communication protocol in the ARPA network, in: APPS Conference Proc., SJCC, 1970, pp. 589–597.
[10] S.M. Ornstein, F. Heart, W. Crowther, H. Rising, S. Russell, A. Michel, The Terminal IMP for the ARPA, Computer Network, pp. 243–254.
[11] H. Frank, R.E. Kahn, L. Kleinrock, Computer Communication Network Design: Experience with Theory and Practice, pp. 255–270.
[12] S.D. Crocker, J.F. Heafner, R.M. Metcalfe, J.B. Postel, Function-Oriented Protocols for the ARPA Computer Network, pp. 271–279.
[13] R.H. Thomas, D.A. Henderson, McROSS: A Multi-Computer Programming System, pp. 281–293.
[14] H. Frank, I.T. Frisch, R. Van Slyke, W. Chou, Optimal design of centralized computer networks, Networks 1 (1) (1971) 43–57.
[15] A. Girard, B. Sansó, Multicommodity flow models, failure propagation, and reliable loss network design, IEEE/ACM Trans Netw (TON) 6 (1) (1998) 82–93.
[16] A. Narula-Tam, E. Modiano, Dynamic load balancing for WDM-based packet networks, in: INFOCOM, 2000, pp. 1010–1019.
[17] A.S. Reaz, V. Ramamurthi, S. Sarkar, D. Ghosal, S. Dixit, B. Mukherjee, CaDAR: an efficient routing algorithm for a wireless-optical broadband access network (WOBAN), J Opt Commun Netw 1 (5) (2009) 392–403.
[18] L. Chiaraviglio, M. Mellia, F. Neri, Minimizing isp network energy cost: formulation and solutions, IEEE/ACM Trans Netw (TON) 20 (2) (2012) 463–476.
[19] L. Fratta, M. Gerla, L. Kleinrock, The flow deviation method: an approach to store-and-forward communication network design, Networks 3 (2) (1973) 97–133.
[20] E. Kometani, T. Sasaki, On the stability of traffic flow, Report No 1, J Oper Res (1958).
[21] H. Frank, W. Chou, Routing in computer networks, Networks 1 (2) (1971) 99–112.
[22] L.R. Ford, D.R. Fulkerson, A suggested computation for maximal multi-commodity network flows, Manage Sci 5 (1) (1958) 97–101.
[23] H. Frank, Research in store and forward computer networks, in: DTIC Document, 1972.
[24] G.B. Dantzig, Linear Programming and Extensions, Princeton University Press, 1965.
[25] A. Charnes, W. Cooper, Multicopy traffic network models, Theory Traffic Flow (1961) 85–96.
[26] S.C. Dafermos, F.T. Sparrow, Traffic assignment problem for a general network, J Res Natl Bureau Stand Sel B – Math Sci 2 (1969) 91–118.
[27] B. Yaged, Minimum cost routing for static network models, Networks 1 (2) (1971) 139–172.
[28] J.A. Tomlin, Minimum-cost multicommodity network flows, Oper Res 14 (1) (1966) 45–51.
[29] R. Gallager, A minimum delay routing algorithm using distributed computation, IEEE Trans Commun (January) (1977).
[30] R. Gallager, J. Golestani, Flow control and routing algoritms for data networks, in: ICCC80, Atlanta, Georgia, October 1980.
[31] Wooseong Kim, Mario Gerla, NAVOPT: Navigator Assisted Vehicular route OPTimizer, in: IMIS, June 2011.
[32] Chi-Yao Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, R. Wattenhofer, Achieving high utilization with SoftwareDriven WAN, in: ACM SIGCOMM, 2013.
[33] Sumo Simulator. <http://sourceforge.net/apps/mediawiki/sumo/>.
[34] John B. Kenney, Dedicated short-range communications (DSRC), in: IEEE Proceedings, 2011.

Luigi Fratta received the Doctorate in Electrical Engineering from the Politecnico di Milano, Milano, Italy, in 1966. As a Research Assistant at the Department of Computer Science, University of California, Los Angeles, he participated in data network design under the ARPA project from 1970 to 1971. From 1975 to 1976 he was at the Computer Science Department of the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, working on modeling analysis and optimization techniques for teleprocessing systems. In 1979 he was a Visiting Associate Professor in the Department of Computer Science at the University of Hawaii. In the summer of 1981 he was at the Computer Science Department, IBM Research Center, San Josè, CA, working on local area networks. During the summers of 1983, 1989 and 1992 he was with the Research in Distributed Processing Group, Department of Computer Science, U.C.L.A., working on fiber optic local area networks. During the summer of 1986 he was with Bell Communication Research working on metropolitan area networks. In 1994 he has been Visiting Scientist at NEC Network Research Lab, Japan. During the summer of 2000 he was Visiting Professor at Computer Sciences Department of the University of Canterbury, New Zealand. During the winter 2009 he was Visiting Scientist at the EE Dept. Imperial College, UK. During Fall 2010 he was Visiting Scholar at Computer Science Dept., Univ. of California, Los Angeles, USA. From 1980 to 2012 he has been Full Professor at the Dipartimento di Elettronica e Informazione of the Politecnico di Milano. At present he is enjoying his retirement cultivating

research interests on computer communication networks, packet switching networks, multiple access systems, modeling and performance evaluation of communication systems, local area networks, wireless cellular systems and green networking. Dr. Fratta is Fellow of IEEE.



Mario Gerla is a Professor in the Computer Science Dept at UCLA. He holds an Engineering degree from Politecnico di Milano, Italy and the Ph.D. degree from UCLA. He became IEEE Fellow in 2002. At UCLA, he was part of the team that developed the early ARPANET protocols under the guidance of Prof. Leonard Kleinrock. He joined the UCLA Faculty in 1976. At UCLA he has designed network protocols for ad hoc wireless networks (ODMRP and CODECast) and Internet transport (TCP Westwood). He has lead the ONR MINUTEMAN project, designing the next generation scalable airborne Internet for tactical and homeland defense scenarios. His team is developing a Vehicular Testbed for safe navigation, content distribution, urban sensing and intelligent transport. He serves on the IEEE TON Scientific Advisory Board. He was recently recognized with the annual MILCOM Technical Contribution Award (2011) and the IEEE Ad Hoc and Sensor Network Society Achievement Award (2011).



Leonard Kleinrock is considered a father of the Internet, having developed the mathematical theory of packet networks, the technology underpinning the Internet, while an MIT graduate student. This was in 1962, many years before the birth of the Internet which occurred in his laboratory when his UCLA Host computer became the first Internet node in September 1969. A month later, he directed the transmission of the first message ever to pass over the Internet. He wrote the first paper and published the first book on the subject.

In his early research, he set up the model of computer networks, identified queueing theory as the key network evaluation tool and developed optimal design procedures. He introduced the concept of packetizing messages, evaluated the effect on performance of topology and routing procedure and articulated the underlying principles of data networks, identifying the importance of resource sharing, the economy of scale, and the use of distributed control.

Dr. Kleinrock received his Ph.D. from MIT in 1963. He currently serves as a Distinguished Professor of Computer Science at UCLA. He has published over 250 papers and authored six books on a wide array of subjects including packet switching networks, packet radio networks, local area networks, broadband networks, nomadic computing, peer-to-peer networks, and intelligent software agents.

Dr. Kleinrock is a member of the National Academy of Engineering, the American Academy of Arts and Sciences, an IEEE fellow, an ACM fellow, an INFORMS Fellow, and an IEC fellow. Among his many honors, he is the recipient of the Ericsson Prize, the NAE Draper Prize, the Marconi Prize, the Okawa Prize, and was further recognized when he received the 2007 National Medal of Science, the highest honor for achievement in science bestowed by the President of the United States.