Optimal Routing in a Packet-Switched Computer Network

DAVID G. CANTOR, SENIOR MEMBER, IEEE, AND MARIO GERLA

Abstract—The problem of finding optimal routes in a packet-switched computer network can be formulated as a nonlinear multicommodity flow problem.

The application of traditional mathematical programming techniques to the solution of the routing problem for reasonably large networks is computationally inefficient. Satisfactory results have been recently obtained with various heuristic techniques; however, such techniques are nonoptimal and subject to several limitations.

The purpose of this paper is to present a method, based on decomposition techniques, which is exact and, at the same time, computationally very efficient. Such a method, originally developed for a computer network application, can be extended to a variety of convex multicommodity flow problems.


I. INTRODUCTION

The optimal routing problem in a computer network consists of the determination of the optimal routing policy, i.e., the set of routes on which packets have to be transmitted in order to optimize a well-defined objective function (e.g., delay, cost, throughput etc.). Under appropriate assumptions, the optimal routing problem can be formulated as a nonlinear multicommodity flow problem [1].

General techniques for solving multicommodity problems can be found in the mathematical programming literature [2], [3]; however, the straightforward application of these techniques to the routing problem in computer networks proves to be computationally cumbersome. In fact, the algorithms for the determination of optimal topology and channel capacities in a computer network require hundreds of optimal routing computations; therefore, an extremely fast routing technique has to be used. For that reason, considerable effort has been spent in developing heuristic techniques [1], [4]. Quite satisfactory results have been obtained and computational efficiency has been greatly improved; however, all of these techniques are affected by various limitations.

In this paper the problem is approached via mathematical programming. The constraint equations are first investigated and some interesting properties are recognized. Taking advantage of these properties, a decomposition method is applied, in a greatly simplified form, and an algorithm for the exact solution is presented. The algorithm is shown to be computationally competitive with the existing heuristic techniques.

II. THE ROUTING PROBLEM

Consider a packet-switched (also referred to as store-and-forward) computer communication network [5]. In such a network, messages are segmented into packets, and each packet traveling from source $N_i$ to destination $N_j$ is "stored" in a queue at each intermediate node $N_b$, while awaiting transmission, and is sent "forward" to $N_i$, the next node in the route from $N_i$ to $N_j$, when channel $(k,l)$ is free. Thus, at each node there are several queues, one for each output channel. Packet flow requirements between nodes arise at random times and packets are of random length; therefore, channel flows, queue lengths, and packet delay are random variables.

Under appropriate assumptions, it is possible to relate the average delay $T$ of a packet traveling from source to destination (the average is over time and over all pairs of nodes) to the average flows in the channels. The result of the analysis is [5]

$$T = \left(1/\gamma\right) \sum_{i=1}^{NA} f_i / (C_i - f_i)$$

where

- $T$ total average delay per packet [seconds/packet].
- $NN$ number of nodes, $NA = \text{number of arcs}$.
- $r_{ij}$ average packet rate from source $i$ to destination $j$ [packet/second].
- $\gamma = \sum_{i=1}^{NN} \sum_{j=1}^{NN} r_{ij}$ total packet arrival rate from external sources (throughput) [packet/second].
- $f_i$ total bit rate on channel $i$ [bits/second].
- $C_i$ capacity of channel $i$ [bits/second].

The expression of $T$ becomes more complicated when more details are included in the model [6], [7]; the method proposed here applies also to those more general models.

1. One can distinguish between routing policies which are determined a priori and are time invariant (deterministic policies), and policies which vary in time, according to load and queue fluctuations (adaptive policies) [9]. Here we restrict the analysis to deterministic policies.

Assumptions: Poisson arrivals at nodes, exponential distribution of packet length, independence of arrival processes at different nodes, independence assumption of service times at successive nodes [5].

$fi$ is given by the contribution of all packets transmitted over channel $i$. 
The routing problem for a packet-switched communication network is formulated as follows:

**Given**: a directed network of \(NN\) nodes and \(NA\) arcs, an \(NN \times NN\) matrix \(R = [r_{ij}]\), called the requirement matrix, whose entries are nonnegative.

Minimize (over \(f\)) \(T = (1/\gamma) \sum \sum a_{ij} f_{ij} / (C_i - f_i)\) where \(f = (f_1, f_2, \ldots, f_{NN})\).

**Constraints**: 1) \(f\) is a multicommodity flow satisfying the requirement matrix \(R\). 2) \(f \leq C\) where \(C = (C_1, C_2, \ldots, C_{NN})\).

We next investigate the properties of the set of feasible flows \(F\), defined by constraints 1) and 2).

**III. THE SET OF FEASIBLE FLOW PATTERNS**

**A. Routing Policy Representation**

The most general routing policy can be described by providing, for each \((i,j)\) commodity, a set of routes \(\pi_{ij}^{(k)}\), \(k = 1, \ldots, K_{ij}\) from node \(i\) to node \(j\), associated with some weights \(\alpha_{ij}^{(k)}\), \(k = 1, \ldots, K_{ij}\) \((\alpha_{ij}^{(k)} > 0, \sum_{k} \alpha_{ij}^{(k)} = 1)\); by that we mean that commodity \((i,j)\) is transferred from \(i\) to \(j\) along \(K_{ij}\) routes, and route \(\pi_{ij}^{(k)}\) carries a fraction \(\alpha_{ij}^{(k)}\) of commodity \((i,j)\).

A routing policy can also be expressed by a set of routing tables \(RT_i\), \(i = 1, \ldots, NN\) [4]. The routing table for node \(i\), \(RT_i\) is an \(i \times NN\) matrix where \(i\) is the number of nodes adjacent to node \(i\); the entry \(RT_i(k,d)\) of such a matrix represents what fraction of the traffic that has arrived at node \(i\) and has destination \(d\) must be transmitted to neighboring node \(k\).

The entries of a routing table satisfy the following conditions:

\[0 \leq RT_i(k,d) \leq 1, \sum_{k=1}^{li} RT_i(k,d) = 1.\]

The practical importance of routing tables follows from the fact that each table contains all the information required by the corresponding node for the routing of the incoming traffic.

A very important case of routing policy is the shortest route policy, which transmits packets along shortest routes computed according to a well-defined assignment of lengths to the arcs. Such a policy is simply described by the shortest route matrix \(RM\), an \(NN \times NN\) matrix whose entry \(RM(i,j)\) is the label of the next node on the shortest route to node \(j\) starting from node \(i\) [3].

Another representation of a multicommodity flow consists of the vector \(f\) of the flows in all channels (recall that \(f_i\) is the total flow in channel \(i\), the sum of all commodities flowing through \(i\)). It can be easily seen that \(f\) does not completely characterize a multicommodity flow. For instance, two different sets of routes might yield the same \(f_i\) in the solution of problem (2), however, such sets of routes can be considered equivalent because they give the same value of the objective \(T\). It is of interest, therefore, to investigate the properties of the feasible set \(F = F_a \cap F_b\) where \(F_a\) = \{ \(f\) satisfies multicommodity constraint 1\}, and \(F_b\) = \{ \(f\) satisfies capacity constraint 2\}.

**B. Multicommodity Constraints**

Let \(s_{ij}^{(m,n)}\) be the flow in arc \((i,j)\) due to commodity \((m,n)\). For the conservation of the flow at each node we have

\[\sum_{j=1}^{NN} s_{ij}^{(m,n)} - \sum_{j=1}^{NN} s_{ij}^{(m,n)} = \begin{cases} -r_{mn}, & \text{if } i = m \\ 0, & \text{if } i \neq m,n \\ +r_{mn}, & \text{if } i = n \end{cases}\]

for \(i = 1, \ldots, NN\). For nonnegativity of the flows we have

\[s_{ij}^{(m,n)} \geq 0, \quad \text{for all } i,j.\]

Let \(s^{(m,n)} = (s_{ij}^{(m,n)})\) be the flow due to commodity \((m,n)\) in the arc labeled \(i\) and let \(S^{(m,n)} = \{s_{ij}^{(m,n)} | \text{satisfies (3) and (4)}\}\). Then \(S^{(m,n)}\) is defined by linear equations and inequalities, is a convex polyhedron [2]. If we restrict our consideration to those flows \(s^{(m,n)}\) that can be represented as a convex combination of loopless paths, then the set \(S^{(m,n)}\) is also closed and bounded.

By definition, any multicommodity flow \(f\) satisfying requirement matrix \(R = (r_{mn})\) can be expressed as follows:

\[f = \sum_{m=1}^{NN} \sum_{n=1}^{NN} s^{(m,n)}.\]

Therefore, the set of multicommodity flows \(F_a\) is also a convex, closed, and bounded polyhedron. The flows corresponding to the extreme points of \(F_a\) have the following interesting property: they are shortest route flows [9]. Conversely, one can easily show that all shortest route flows correspond to extreme points of \(F_a\). In the sequel we refer to such flows as extremal flows (see Fig. 1).

Since \(F_a\) is a convex polyhedron, any \(f \in F_a\) can be expressed as a convex combination of extremal flows \(\phi^i, i = 1, \ldots, r\) where \(r\) is the total number of extremal flows in \(F_a\):

\[f = \sum_{i=1}^{r} \lambda_i \phi^i, \sum_{i=1}^{r} \lambda_i = 1, \lambda_i \geq 0, \quad i = 1, \ldots, r.\]

It can be shown that an optimal routing policy does not allow loops [8].

A shortest route flow is a multicommodity flow whose routes can be described by a shortest route matrix, computed for a well-defined assignment of lengths \([l_i]\) to the arcs. It can be shown that such a flow minimizes the linear objective \(2f_i\) [3].
C. Capacity Constraints

The set $F_b = \{ f \mid f \leq C \}$ is a convex set; hence the feasible set $F = F_a \cap F_b$ is also convex.

Notice that

$$\lim_{t_i \to 0^+} T(f) = +\infty, \quad i = 1, \cdots, NA.$$

This corresponds to the existence of barriers that prevent the solution from approaching the boundary of $F_b$ when $T$ is minimized. Using the mathematical programming terminology, we can say that the objective function incorporates the capacity constraints as interior penalty or barrier functions.

This property is very important from a practical point of view because it guarantees automatically the constraint 2) feasibility during the application of the usual nonlinear optimization techniques, once a feasible flow $f_0$ is found; it allows disregarding constraint 2) and treating the routing problem as an unconstrained multicommodity flow problem.

IV. A DECOMPOSITION APPROACH

A. New Formulation

The objective function of (1) is strictly convex (as the sum of convex functions) and the set $F$ is closed, bounded, and convex: therefore, if $F$ is not empty, one and only one local minimum exists, and it is also the global minimum [10]. In order to find the minimum, we apply the decomposition method [2] and reformulate problem (2) as follows, using (6):

$$\min T = \frac{1}{\gamma} \sum_{k=1}^{NA} \left[ \sum_{i=1}^{r} \lambda_k \phi_i^{(k)}(C_i) \right] / \left( C_i - \sum_{k=1}^{\gamma} \lambda_k \phi_i^{(k)} \right)$$

subject to

$$\sum_{k=1}^{r} \lambda_k = 1$$

and

$$\lambda_k \geq 0, \quad k = 1, \cdots, r$$

where

$$\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_r).$$

The constraint equations in (7) are much simpler than those in (2); on the other hand, the number of variables is enormous, and in principle, the knowledge of all extremal flows is required.

However, the effectiveness of the decomposition approach is based on the following facts.

1) Any flow $f$ can be represented by, at most, $NA + 1$ extremal flows; thus, we have to consider no more than $NA + 1$ variables $\lambda_k$ at a time.

2) There exists a procedure that sequentially generates only the $\phi_i^{(k)}$’s that produce improvement of the objective function.

The fundamental steps of the decomposition method are discussed in the following sections.

B. Maximum Number of Extremal Flows

Fact: Any multicommodity flow $f$ can be expressed as a convex combination of $\leq NA + 1$ extremal flows (where $NA$ = number of arcs).

Proof: Recall from (6) that any $f$ can be expressed as a convex combination of $all$ extremal flows $\phi^{(i)}$, $i = 1, \cdots, r$:

$$\phi_1^{(i)} \lambda_1 + \phi_2^{(i)} \lambda_2 + \cdots + \phi_r^{(i)} \lambda_r = f_i$$

$$\phi_N^{(i)} \lambda_1 + \cdots + \phi_N^{(i)} \lambda_r = f_N$$

$$\lambda_i \geq 0, \quad i = 1, \cdots, r.$$

By assumption, there exists a feasible solution $\lambda$ to (8); therefore, there exists, from the theory of linear programming [2], a basic feasible solution, i.e., a solution in which at most $m$ variables are nonzero, where $m$ is the rank of the system’s matrix. In our case $m \leq NA + 1$; therefore, given any flow $f$, we can always find a solution $\lambda$ with $\leq NA + 1$ nonzero components, i.e., we can express $f$ as a convex combination of $\leq NA + 1$ extremal flows.

Q.E.D.

As a consequence, if $f$ is expressed in terms of $b > NA + 1$ extremal flows, we can always eliminate $b - (NA + 1)$ components from the convex combination. In particular, if $b = NA + 2$, we need to eliminate only one extremal flow; such an elimination is obtained with the pivot step, a well-known operation in the theory of linear programming [2].

C. Conditions for Optimality

From the Kuhn–Tucker theorem [10], a solution $\lambda$ to problem (7) is optimal if and only if

$$\sum_{k=1}^{r} \lambda_k = 1, \lambda_k \geq 0, \quad k = 1, \cdots, r$$

(9a)

$$\frac{\partial T}{\partial \lambda_k} \geq \mu_0 \text{ if } \lambda_k = 0$$

(9b)

where $\mu_0$ is a constant.

*Similar conditions were stated by Dafermos in [11].
Let
\[ \mu_k = \frac{\partial T}{\partial \lambda_k} = \sum_{i=1}^{N_a} l \phi_i^{(k)} \]  
(10)

where
\[ l_i = \frac{\partial T}{\partial f_i} = \frac{C_i}{\gamma (C_i - f_i)^2}. \]

Then \( \mu_k \) can be interpreted as the total cost of flow \( \phi^{(k)} \), assuming cost \( l_i \) per unit of flow in arc \( i \); physically, \( \mu_k \) represents the increment of delay \( T \) per unit increment of \( \lambda_k \).

We can rewrite (9b) as follows:
\[ \mu_k = \begin{cases} \mu_0, & \text{if } \lambda_k > 0 \\ \geq \mu_0, & \text{if } \lambda_k = 0 \end{cases} \quad \forall k. \]  
(11)

Condition (11) has a very simple physical interpretation: all nonzero extremal flows must have the same cost \( \mu_0 \) (i.e., the same incremental delay), and any extremal flow that has cost \( > \mu_0 \) must be at zero level. Using (9a), an equivalent formulation of (11) is the following:
\[ \sum_\gamma \lambda_\gamma \mu_\gamma = \min_k \mu_k \]  
(12)

or, using (6) and (10),
\[ \sum_{i=1}^{N_a} l f_i = \min \{ \sum_{i=1}^{N_a} \lambda \phi_i^{(k)} \}. \]  
(13)

Recalling that the solution of a linear min cost flow problem is always an extremal flow [2], we rewrite (13) as follows:
\[ \sum_{i=1}^{N_a} l f_i = \min \{ \sum_{i=1}^{N_a} l \phi_i \}. \]  
(14)

Condition (14) simply states that \( f \) is optimal if and only if there exists no flow \( v \) that provides a delay increment less than \( f \). In order to verify (14), only one shortest route computation is required.

D. Master Problem

Suppose that we have a restricted set of extremal flows \( \phi^{(k)} \), \( k = 1, \ldots , b < r \), and we want to minimize (7) over the subset of variables \( \lambda_\gamma \) (restricted master problem). This problem can be solved using the gradient projection method [10], a constrained optimization method in which the gradient \( \nabla_f T \), given by (see (10))
\[ \nabla_f T = (\mu_1, \mu_2, \ldots , \mu_b) \]
is projected on the hyperplane \( \sum \lambda_i - 1 = 0 \), and the components of \( \nabla_f T \) that point towards unfeasible directions are set to zero. In this way, a feasible downhill direction \( - \nabla_f T \) is obtained. At each iteration of the gradient projection method, the following one-dimensional minimization over \( x \) is performed:
\[ \min_{x>0} T (\sum_{i=1}^{b} (\lambda_i + \alpha [- \nabla_f T]_i) \phi^{(i)}) \]

where \([ - \nabla_f T]_i \) is the \( i \)-th component of \(- \nabla_f T \) and where \( \alpha \), for nonnegativity of \( \lambda \), must satisfy the following inequality:
\[ \alpha \leq \lambda_i / [ - \nabla_f T]_i \]  
when \([ - \nabla_f T]_i > 0 \), \( i = 1, \ldots , b \).

As in the usual gradient method, the computation of \(- \nabla_f T \) and the one-dimensional optimization are repeated until \( || \nabla_f T || = 0 \). When this happens, condition (11) for optimality is satisfied (at least over the restricted set of extremal flows), and therefore the restricted master problem is solved.

The next step consists of testing whether the optimal solution to the restricted master problem is also the optimal solution to the global problem.

E. The Flow Generating Subproblem

Suppose we have solved the restricted master problem, i.e., we have found a solution \( \lambda = (\lambda_1, \lambda_2, \ldots , \lambda_b) \) satisfying (11). In order to verify global optimality, we solve the following linear min cost flow problem (the flow generating subproblem):
\[ \min \sum_{i=1}^{N_a} l \phi_i \]  
(15)

Let \( \phi' \) be the solution of (15) and let
\[ \mu' = \sum_{i=1}^{N_a} l \phi_i' \]  
If \( \mu' \geq \mu_0 = \sum_{i=1}^{N_a} l f_i \), the solution \( \lambda \) is a global optimum from condition (14). If, on the other hand, \( \mu' < \mu_0 \), \( \lambda \) is not a global optimum; improvement can be obtained by solving a new restricted master problem, which includes \( \phi' \).

V. THE EXTREMAL FLOWS (EF) METHOD

The EF method consists of the repeated solution of the master problem and the subproblem until the required accuracy on the value of the delay \( T \) is obtained. Typically, a new extremal flow is generated by the subproblem at each iteration; when several iterations are required, the number of current extremal flows is kept \( \leq N_A + 1 \) by the pivot operation. The following is an outline of the EF algorithm.

**EF Algorithm:** (16)

**Step 0—Initialization:**

\[ b = N_A + 1 \text{: number of current extremal flows.} \]

\[ \phi^{(0)} = (\phi^{(1)}, \ldots , \phi^{(b)}) \text{: initial set of extremal flows.} \]

\[ \lambda^{(0)} = (\lambda_1^{(0)}, \ldots , \lambda_b^{(0)}) \text{: initial basic solution.} \]

\[ f^{(0)} = \sum_{i=1}^{b} f^{(i)} \text{: initial feasible flow.} \]

\[ n = 1 \]

**Step 1—Master Problem:**

\[ \frac{\partial T}{\partial \lambda} = \sum_{i=1}^{N_a} l \phi_i \]  
where \([ - \nabla_f T]_i \) is the \( i \)-th component of \(- \nabla_f T \) and where \( \alpha \), for nonnegativity of \( \lambda \), must satisfy the following inequality:
\[ \alpha \leq \lambda_i / [ - \nabla_f T]_i \]  
when \([ - \nabla_f T]_i > 0 \), \( i = 1, \ldots , b \).

It should be mentioned that in some situations the straightforward gradient projection method application does not converge to an optimum solution; in such cases the method must be properly modified so as to include antizigzagging precautions [20].
Minimize $T(f)$ over the restricted set of extremal flows $\Phi^{n-1}$. At the end of such minimization, let

$$\lambda^n = (\lambda_1^n, \lambda_2^n, \ldots, \lambda_N^n): \text{optimal solution of the restricted master,}$$

$$f^n = \sum_{k=1}^b \phi^{(k), n-1} \cdot \lambda_k^n: \text{optimal flow for the restricted master,}$$

$$l^n = (l_1^n, l_2^n, \ldots, l_{NA}^n): \text{vector of equivalent lengths.}$$

**Step 2—Pivot Step:**
If $b = NA + 1$, let $\Phi^n \leftarrow \Phi^{n-1}$ and go to Step 3. If $b = NA + 2$, eliminate, with a pivot operation, one of the extremal flows. After the elimination, let

$$\lambda^n = (\lambda_1^n, \lambda_2^n, \ldots, \lambda_{NA+1}^n): \text{new basic solution,}$$

$$\Phi^n = \text{new basis.}$$

**Step 3—Subproblem:**
Compute $\varphi^n$, the shortest route flow corresponding to the metric $F^n$.

**Step 4—Stopping Rule:**
Let

$$\theta^n = \sum_{i=1}^{NA} l_i^n (f_i^n - \varphi_i^n).$$

If $\theta^n < \epsilon$, where $\epsilon$ is a positive tolerance, stop: $f^n$ is optimal within such a tolerance.\(^{10}\) Otherwise go to Step 5.

**Step 5:** Let

$$b = NA + 2,$$

$$\varphi^{(NA+2), n} \leftarrow \varphi^n,$$

$$\lambda_{NA+2}^n = 0,$$

$$n = n + 1,$$

Go to Step 1.

At the end of the algorithm we have the optimal flow $f$, expressed as a convex combination of at most $NA + 1$ extremal flows. To each extremal flow there is associated a routing matrix; from all such routing matrices, and from $\lambda$, we can obtain the routing tables for $f[8]$. \(^{11}\)

**VI. CONVERGENCE OF THE EF ALGORITHM**

**Theorem:** The EF algorithm applied to problem (2) generates a sequence $f^n$ which converges to the optimal solution $f^*$.\(^{12}\)

**Proof:** First notice that the constraint set $F$ is closed and bounded; $T(f)$ is strictly convex and bounded below $[T \geq 0 \text{ from (1)}]$; $\partial T/\partial f_i$ is continuous and $\partial^2 T/\partial f_i^2$ is bounded above by a positive $M < \infty$, for all $f \in F_a \cap F'_s$, where $F'_s = \{ f | f \leq C - \epsilon \}$, and $\epsilon$ is $> 0$.\(^{11}\) From the above properties, it follows that the delay improvement obtained in Step 1 of the EF algorithm, $\Delta T(f^n) = | T(f^{n+1}) - T(f^n) |$, is bounded below by the following expression\(^{12}\) [8]:

$$\Delta T(f^n) \geq \frac{\theta^n}{2} \min \left\{ \left( \frac{q_i}{q_i} \right)^2, 1 \right\}$$

where

$$q_i = M \sum_{i=1}^{NA} (f_i^n - \varphi_i^n)^2$$

$$\theta^n = \sum_{i=1}^{NA} l_i^n (f_i^n - \varphi_i^n)$$

$$\varphi^n = \text{shortest route flow at iteration } n.$$\(^{10}\)

Now, since $T(f^n)$ is decreasing, $\lim_{n \to \infty} T(f^n) = \bar{T} \geq 0$ exists; hence $\lim_{n \to \infty} \Delta T(f^n) = 0$ and by (17) $\lim_{n \to \infty} \theta^n = 0$. From footnote 10 it follows that $\bar{T}$ is the solution to our problem and since $F$ is compact and $T$ is strictly convex, the $f^n$ converge to a limit $f^*$. Notice that the accuracy of the master problem solution is not critical to the convergence, although it clearly affects the rate of convergence. The exactness of the subproblem solution (i.e., min cost flow), on the other hand, is essential to convergence. When selecting the number of gradient iterations to be performed within each master problem optimization, one must evaluate the delay improvement and the computational effort of an additional gradient iteration, as compared to the delay improvement and the computational effort produced by a new shortest route computation. In most applications, satisfactory results are obtained when execution time is equally allocated to master problem and subproblem.

**VII. THE STARTING FEASIBLE FLOW**

The technique used here for finding a starting feasible flow consists of relaxing the capacity constraints and introducing penalty functions outside the feasible region.

A modified objective function $T_0$ is introduced (see Fig. 2):

$$T_0(f) = \frac{1}{\gamma} \sum_{i=1}^{NA} T_{oi}(f_i)$$

where

$$T_{oi}(f_i) = \begin{cases} T_i(f_i), & f_i \leq f_{oi} \\
T_i(f_{oi}) + \left[ \frac{\partial T}{\partial f_i} \right]_{f_{oi}} (f_i - f_{oi}), & f_i > f_{oi}
\end{cases}$$

$$f_{oi} = (1 - \alpha)C_i, \quad \alpha < 1.$$\(^{11}\)

After such a modification, the capacity constraint is relaxed, the feasible region is enlarged to all $F_a$, and any extremal flow is feasible. In particular, we can choose the extremal flow $\phi^{(0)}$ corresponding to the metric $\{l_{oi}\}$ where

$$l_{oi} = \left[ \frac{\partial T}{\partial f_i} \right]_{f_{oi}} = \frac{1}{\gamma C_i}.$$\(^{11}\)

The EF algorithm can be initiated, using the modified objective function $T_0(f)$.\(^{12}\)

If the problem has a solution and if $\alpha$ is small enough, after a few iterations, say at iteration $k$, we have
Then \( f^{(k)} \in P \), and we can continue the application of the algorithm using the original objective \( T(f) \).

VIII. EXTENSIONS

The EF method, here introduced in the context of a packet-switching computer communication problem, is, in fact, very general and can be used to determine the optimal routing for a variety of network flow applications (communication, transportation [11], distribution problems, etc.).

In particular, the EF method in the form presented in Section V can be applied to any min "cost" flow problem such that

1) \( P(f) \) is convex,
2) the cost \( P(f) \) depends only on the total flow in each arc,
3) \( \partial P/\partial f_i \) is continuous and nonnegative,\(^{13}\)
4) there are no additional constraints unless they may be included in \( P(f) \) as penalty functions.

More generally, slight modifications to the EF method enable relaxation of condition 2) and application of the method to problems in which there are \( n \) different classes of "customers," each class contributing differently to the total cost (i.e., \( P = P(f^{(1)}f^{(2)}, \ldots, f^{(n)}) \) where \( f^{(i)} \) is the flow pattern corresponding to class \( i \)). Such classes could represent, for example, different types of vehicles in a transportation network [12], or messages with different priorities in a computer communication network [13]. Without discussing the details, we just mention that, in an EF algorithm properly modified to include multiple classes,

![Fig. 2. Modified objective function for finding the starting feasible flow.](image)

1) for each class \( i \) we need a distinct basis \( \Phi^{(i)} \), a basic solution \( \lambda^{(i)} \), a flow \( f^{(i)} \), \( i = 1, \ldots, n \);
2) at each EF iteration we perform \( n \) pivot steps, one for each class;
3) at each EF iteration we solve \( n \) min cost flow subproblems:

\[
\min \sum_{k=1}^{NA} f^{(k)} \quad i = 1, \ldots, n
\]

and perform \( n \) optimality test:

\[
\sum_{k=1}^{NA} f^{(k)} - \varphi^{(i)}(f) < \epsilon, \quad i = 1, \ldots, n;
\]

4) in the solution of the master problem, the downhill direction \(- \nabla P\) is defined as follows:

\[
- \nabla P' = (- \nabla P^{(1)}, - \nabla P^{(2)}, \ldots, - \nabla P^{(n)})
\]

where \(- \nabla P^{(i)}\) is the usual downhill direction for class \( i \).

Notice that the introduction of different classes produces a formal, rather than conceptual, complication of the EF algorithm. Next, considering condition 4), it is shown in [8] that the EF method can also be extended to convex min cost flow problems with concave, nonnegative constraints.

IX. APPLICATION: OPTIMAL ROUTING FOR THE ARPA COMPUTER NETWORK

The ARPA computer network is an experimental packet-switched network connecting several computer facilities in the United States. A detailed description of the network is given in [15]–[18], [6], [19]. One of the topologies recently proposed for the ARPA network is given in Fig. 3; for this topology, the routing problem is solved here. The channel capacities are all 50 (kbits/second). The requirement matrix \( R \) is assumed uniform \( r_{ij} = r \) for all \( i \neq j \); \( r_{ii} = 0 \). The efficiency of the system is measured in terms of the average delay \( T \) where \( T \) now includes acknowledgment traffic and propagation delay [6]:

\[
T = \sum_{i=1}^{NA} \frac{1}{C_i} (\mu'/\mu - 1) + \frac{1}{C_i - f_i} + \frac{1 + 5d_i/\beta}{\beta}
\]

where

\[
1/\mu = \text{average packet length without acknowledgment} \quad \text{(bits/packet)}
\]
\[
1/\mu' = \text{average packet length including acknowledgment} \quad \text{(bits/packet)}
\]
\[
d_i = \text{length of the link (miles)}
\]
\[
\beta = \text{speed of light (miles/second)}
\]

The EF algorithm was used to determine the optimal throughput\(^{14}\) \( TR = NN(NN - 1) \) with the constraint \( T \leq T_{\text{max}} \). Table I shows the admissible values of traffic level \( r \) obtained after each EF iteration, with \( T_{\text{max}} =

\(^{13}\) The nonnegativity of \( \partial P/\partial f_i \) excludes negative cycles, which would cause the failure of the shortest route computation, using metric \( i = \partial P/\partial f_i \). The nonnegativity assumption is very reasonable as, in general, the cost \( P(f) \) increases with the flow in each arc.

\(^{14}\) Notice that the EF algorithm, as formulated in (16), solves the min delay problem; however, a straightforward modification allows solving also the max throughput problem for a given max admissible delay \( T_{\text{max}} \).
0.200 s for the network of Fig. 3. Notice that, after ten iterations, \( r \) is 2 percent within the optimum (number of iterations = \( \infty \)).

The algorithm was coded in Fortran; each EF iteration required approximately 0.250 s execution time on an IBM 360/91.

X. EF ALGORITHM VERSUS HEURISTICS

Assuming that equal execution time is allocated to the master problem and the flow generating subproblem in the EF method, and observing that the heuristic techniques [1], [4] also require a shortest route computation and flow assignment at each iteration, we conclude that the EF method and heuristic techniques are computationwise comparable. Heuristic algorithms are, in general, easier to code and require less memory space; furthermore, they give, in most cases, results which are quite close to optimum [1]. However, it is very easy to construct examples in which they perform very poorly. The EF algorithm, on the other hand, can always be driven to the required precision, as we have upper and lower bounds available at each iteration. Furthermore, the EF algorithm is very flexible; it allows for very general objective function, and can be extended to applications with various classes of "customers."

XI. CONCLUSION

We have presented the EF algorithm, an exact routing algorithm for the determination of min cost, deterministic routing policies.

The EF algorithm was originally developed for a computer network application; however, it can be extended to a variety of convex multicmodity flow problems.

The computational efficiency of the algorithm is comparable to that of the existing heuristics; the choice between EF and heuristics should depend upon memory availability, precision requirements, execution time requirements, etc., relative to the specific application. Typically, the order of computation time required per iteration is between \((NN)^2\) and \((NN)^4\) [22]; therefore, the cost becomes prohibitive for very large networks \((NN > 100)\). In such cases, the EF algorithm should be combined with appropriate decomposition [3], [21] or partitioning [8] techniques.

REFERENCES

David G. Cantor (S’53–SM’70) was born in London, England, on April 12, 1935. He received the B.S. degree in physics from the California Institute of Technology, Pasadena, in 1956, and the Ph.D. degree in mathematics from the University of California, Los Angeles, in 1960.

From 1960 to 1962 he was an Instructor at Princeton University, Princeton, N. J., and from 1962 to 1964 he was an Assistant Professor at the University of Washington, Seattle. Since 1964 he has been a faculty member of the University of California, Los Angeles, where he is currently Professor of Mathematics and Computer Science. His interests include network theory and switching networks.

Prof. Cantor is a member of the Association for Computing Machinery, the Society for Industrial and Applied Mathematics, the Mathematical Association of America, and the American Mathematical Society.

Mario Gerla was born in Milan, Italy, on February 21, 1943. He received the Doctorate in Engineering from the Politecnico di Milano, Milan, Italy, in 1968, and the M.S. and Ph.D. degrees in engineering from the University of California, Los Angeles, in 1970 and 1973, respectively.

From 1966 to 1968 he was an Officer in the Italian Navy, and was involved in a NATO experiment on Tactical Satellite Communications. In 1969 he was with the Advanced Systems Section of the GTE subsidiary in Milan. In September 1969 he was awarded a Fulbright scholarship and came to the University of California, Los Angeles, as a graduate student, where he participated in the ARPA Network Project and engaged in research on computer network design. Since January 1973 he has been with the Network Analysis Corporation, Glen Cove, N. Y., where he is involved in several computer network projects and presently is Manager of Computer Networking.

Dr. Gerla is a member of the Institute of Management Science.

Procedures for Eliminating Static and Dynamic Hazards in Test Generation

MELVIN A. BREUER, SENIOR MEMBER, IEEE, AND R. LOYD HARRISON, MEMBER, IEEE

Abstract—One problem associated with test generation algorithms for sequential circuits is that they often produce tests which, when applied to the circuit under test, create static and/or dynamic hazards which may invalidate the test. Usually, for static hazards, but not dynamic hazards, these situations can be predicted using a logic simulator. In this paper we present procedures which can be added to path sensitization test generation algorithms so that the resulting procedure will not produce tests which will be invalidated due to hazards. Incidental to this work is a new simulation technique for handling both static and dynamic hazards. The principal concepts behind this work deal with detecting when hazards are created in a circuit; propagating hazard status information related to a signal line through a circuit; detecting those conditions at flip-flop inputs which necessitate hazard free conditions; and finally, selecting test inputs so that all hazard free conditions are satisfied.

Index Terms—Dynamic hazards, fault detection, simulation, static hazards, test generation for sequential circuits.

I. INTRODUCTION

Most computer programs which generate test sequences for sequential logic networks consist of a few main program modules, one of which is a pattern generator and another which is a logic simulator. The pattern generator determines test vector sequences based upon one or more procedures such as random pattern generation, the D-algorithm [1], [2], Boolean difference [3], or the concept of equivalent normal form [4]. These tests are then processed through a multi-valued fault simulator which determines the set of all faults detected by the test as well as detecting race and hazard conditions [5], [6]. Tests are often found to be invalid since they create races and hazards; hence, the faults supposedly detected by these tests are in reality not detected. Many simulators employ Eichelberger’s technique [7] for detecting hazards; and hence, do not detect the existence of dynamic hazards.

Once a pattern generator has attempted to construct a test sequence for a fault, and has failed due to a hazard condition, it typically has no direction on how to proceed to rectify this situation.

In this paper we present methods to be included in an existing pattern generation procedure so that the resulting procedure will not generate "harmful" hazards, i.e., hazards which will invalidate the test.

II. BASIC CONCEPTS

So that the reader can better understand the context in which the procedure for generating hazard free tests is to be applied, we will briefly discuss the mode of testing