LOOPS IN MULTICOMMODITY FLOWS

Robert G. Gallager*
Massachusetts Institute of Technology
Department of Electrical Engineering
and Computer Science
and
Electronic Systems Laboratory

Abstract

Given the traffic flow from each source to each destination in a network and given the aggregate traffic in each link, we want to find if there is any looping of traffic. A careful definition of looping shows that the question is equivalent to whether some of the aggregate link flows can be reduced without increasing any of the others. It is then shown, through the use of duality in linear programming, that an aggregate flow is loop free iff all the traffic follows shortest routes for some assignment of positive lengths to the links.

It is further shown that there is a finite set of these length assignments, dependent only on the topology of the network, such that every shortest route flow is a shortest route flow for one of those special assignments. Finally, it is shown that any loopfree flow can be realized by a routing in which the sum, over all destinations, of the number of alternate route links required to reach that destination, is at most the number of links minus the number of nodes.

Introduction

When the data in a communication network occasionally travel in loops, these loops generate both an unnecessary loss of resources and also an unnecessary increase in delay. Our purpose here is to define a generalized form of looping and to provide both a mathematical basis and heuristic understanding for this phenomena. We do not develop any new routing algorithms here, but the results provide insight into the weaknesses of current algorithms and into potential directions for overcoming these weaknesses.

Consider a network of n nodes and L directed links. We denote the nodes by the integers 1,..., n and denote the links by integer pairs, (i,k) denoting a link from i to k. We assume that the network is connected in the sense that for each pair of nodes i,j, there is some directed path of links, say $((i,k),(k,\ell),(\ell,m),\ldots,(h,j))$ going from i to j. Let $r_i(j)$ denote the time average data, in bits per second, entering the network at node i and destined for some receiver at node j. Similarly, let $f_{ik}(j) \geq 0$ denote the time average, in bits per second, of the traffic from all inputs flowing over link (i,k) with the destination j. Note that link (k,i) is separate from link (i,k) and it is possible to have $f_{ik}(j) > 0$ and $f_{ki}(j) > 0$. We assume that all traffic destined for j is removed from the network when it gets to j. Thus no traffic for j flows outward from j and $f_{ik}(j)$ is defined only for $i \neq j$, $(i,k)\in\mathscr{L}$, where $\mathscr{L}$ denotes the set of links in the network.

We assume that all traffic destined for j eventually gets there by travelling over paths of the network. Thus all the traffic for j that comes into each node $i \neq j$ must go out of it again, giving us the fundamental conservation equation for each $i \neq j$,

$$r_i(j) + \sum_{\ell} f_{\ell i}(j) = \sum_{k} f_{ik}(j) \tag{1}$$

The first sum above is over the integers ℓ for which $(\ell,i)\in\mathscr{L}$ and $\ell \neq j$. The second is over the integers k for which $(i,k)\in\mathscr{L}$; for brevity, we regard such restrictions as understood in what follows. Also for brevity, we shall denote the set $\{r_i(j)\}$ of inputs over all i,j, $i \neq j$ by the vector r (the ordering of the components is unimportant) and the set $\{f_{ik}(j)\}$ of flows for all $(i,k)\in\mathscr{L}$ and all $j \neq i$ by the vector f.

> Definition: A set of flows $f \geq 0$ is a feasible multicommodity flow for the inputs $r \geq 0$ if (1) is satisfied.

We regard each destination j here as corresponding to a commodity; an amount $r_i(j)$ of that commodity enters the network at node i and is moved to j in accordance with the flow vector f. Each such feasible multicommodity flow represents a particular way of routing the inputs r to their destination. Such routings can be implemented, in the presence of dynamic variations on the inputs, by taking the dynamically varying traffic arriving

*This work was supported in part by ARPA under Grant ONR/N00014-75-C-1183 and NSF under Grant NSF/ENG76-24447.

at i destined for j and allocating it to outgoing links in the same ratio as given by the time average flows $f_{ik}(j)$.

Sometimes one wishes to consider multicommodity flows in which more than one node is used as a sink for a given commodity. This situation can be reduced to the one just described by adding an extra node to the network for each commodity and adding a link from each sink for a commodity to the special node for that commodity.

For purposes of studying the delay experienced by traffic in a network (or some other kind of cost experienced by the commodities), we are often interested primarily in the aggregate, or total flows, in the network lines. In terms of a multicommodity flow f, the aggregate flow F with components F_{ik} is given by

$$F_{ik} = \sum_{j\neq i} f_{ik}(j) \qquad ; \ (i,k)\varepsilon\mathscr{L} \qquad (2)$$

Definition: F is a feasible aggregate flow for an input F if (2) is satisfied for some feasible multi-commodity flow f.

There appears to be no very simple test for whether or not F is a feasible aggregate flow and in fact, the problem is almost identical to the following classic "multicommodity flow problem". Given a set of capacities $C_{ik} \geq 0$, $(i,k)\varepsilon\mathscr{L}$, determine whether there is a feasible multicommodity flow f for which

$$\sum_j f_{ik}(j) \leq C_{ik} \qquad ; \ (i,k)\varepsilon\mathscr{L} \qquad (3)$$

In particular, it should be stressed that conservation of aggregate flow, as expressed by (4) below, is necessary, but far from sufficient, for F to be feasible.

$$\sum_k F_{ik} - \sum_\ell F_{\ell i} = \sum_j r_i(j) - \sum_m r_m(i); \quad \text{all } i \qquad (4)$$

Loops and Loopfreedom in Network Flows

Before defining what we mean by loops in network flows, we give several examples in Figure 1. To avoid cluttering the figures, the links are not shown and each line between two nodes indicates a unit flow. The flows are broken down into individual source-destination flows indicated by the extensions of the lines through the nodes. Figure 1a shows the most obvious type of looping where the traffic from node 1 to 3 loops from 1 to 2 and back before going to 3. Figure 1b shows a situation in which neither the source-destination traffic from node 1 to 3 nor that from 2 to 3 has a loop. However, if we look at the traffic for node 3 as a commodity, we have $f_{12}(3) = f_{23}(3) = f_{21}(3) = f_{13}(3) = 1$. This could also be broken into individual source destination pairs by having the traffic from source 2 to 3 go directly over link (2,3) and the traffic from source 1 to 3 go over the looping path of Figure 1a. Thus we say that the single commodity traffic to node 3 for Figure 1b contains a loop. In general, the single commodity traffic to node j contains a loop if there is a cycle of nodes, $i,k,\ell,\ldots,r,s,i$ such that $f_{ik}(j) > 0$, $f_{k\ell}(j) > 0,\ldots,f_{rs}(j) > 0$, $f_{si}(j) > 0$. Loops of this type have been treated in [1].

Examples of Loops

Figure 1

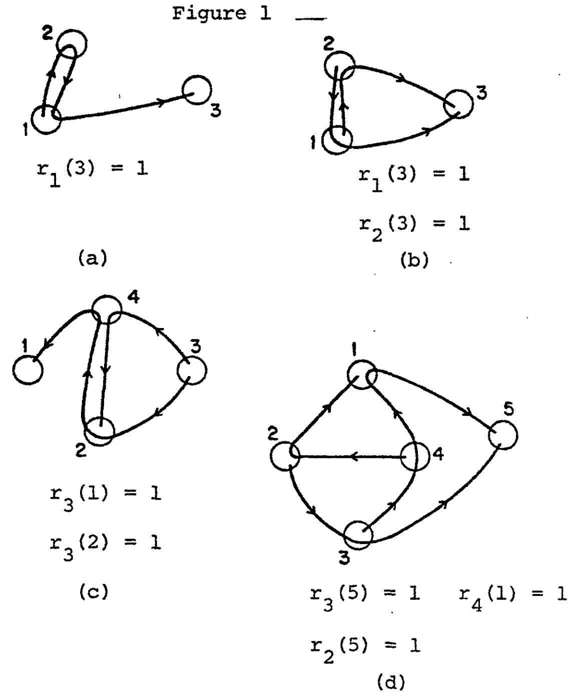
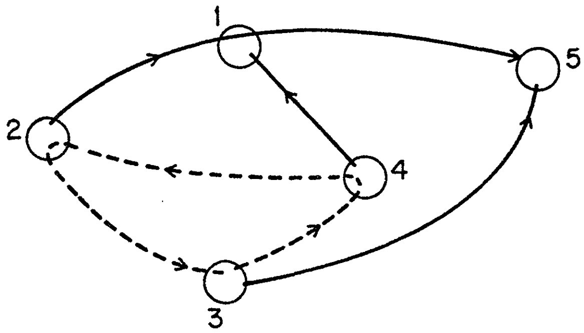


Figure 1c is similar, but harder to handle, since the single commodity flow to destination 1 is loop free and that to destination 2 is loop free. However, the same aggregate flow is achieved by sending the traffic from 3 to 2 directly and that from 3 to 1 via node 4 with a loop to 2. This leads us to our general definition of loops for multicommodity flows.

Definition: A feasible aggregate flow F for the input r contains loops if it can be expressed as F = F' + F" where F' is a feasible aggregate flow for r and F" > 0 is a conservative flow. F is loop-free if it does not contain loops.

A flow F is conservative if it satisfies (4) with the right hand side set to zero. Also, by F" > 0 we mean $F''_{ik} \geq 0$ for all $(i,k)\varepsilon\mathscr{L}$ with strict unequality for at least one $(i,k)\varepsilon\mathscr{L}$. We shall use the terminology positive vector to refer to the stronger condition where all components are positive. It is a well known result of graph theory that a conservative flow can be represented as a finite sum of cycles, or loops.

Figure 1d (due to J. Wozencraft) contains no new complications and is included merely to indicate that loopfreedom is not a property that can be checked by inspection. Below we show the separation of F for Figure 1d into F' and F". Note that simply having a loop of aggregate flow does not destroy loopfreedom; consider for example a 2 node network with traffic flowing both ways.



Lemma 1: A feasible aggregate flow F contains loops iff F' is another feasible aggregate flow for the same inputs with $F' < F$.

Proof: Suppose $F' < F$, and let $F'' = F - F'$. Then $F'' < 0$. Since F and F' each satisfy (4), we can subtract these equations from each other, getting

$$\sum_k F''_{ik} - \sum_\ell F''_{\ell i} = 0$$

Thus F" is conservative and F contains loops. Conversely, if F contains loops, there is an $F' < F$ by definition.

A Linear Programming Approach to Loopfreedom

In this section we formulate a linear programming problem to determine whether or not a feasible aggregate flow is loopfree. Our purpose is not to develop a computational procedure but rather to use the known results of linear programming to determine some of the consequences of loopfreedom. Let r be the input to a given network and let F be a given feasible aggregate flow. Let s be a vector of L slack components with one component s_{ik} for each $(i,k)\varepsilon\mathscr{L}$. Let f' be a vector of $(n-1)L$ components with one component $f'_{ik}(j)$ for each $(i,k)\varepsilon\mathscr{L}$, $j \neq i$. The linear programming problem (LPP) is to choose $s \geq 0$, $f' \geq 0$ to form the minimization:

$$\text{OPT} = \min \sum_{(i,k)\varepsilon\mathscr{L}} - s_{ik} \tag{5}$$

subject to the following $L + n(n-1)$ linearly independent constraints

$$s_{ik} + \sum_j f'_{ik}(j) = F_{ik} \quad ; \ (i,k)\varepsilon\mathscr{L} \tag{6}$$

$$\sum_k f'_{ik}(j) - \sum_{\ell\neq j} f'_{\ell i}(j) = r_i(j) \quad ; \ \text{all } i,j,\ i \neq j \tag{7}$$

If $s \geq 0$, $f' \geq 0$ satisfy (6) and (7), they are called a feasible solution to the LPP; if in addition they minimize (5) they are called an optimal feasible solution. Note that if $f' \geq 0$ satisfies (7), it is a feasible multicommodity flow for r. Thus if we define a vector F' with components F'_{ik} satisfying

$$F'_{ik} = \sum_j f'_{ik}(j) \quad ; \ (i,k)\varepsilon\mathscr{L}, \tag{8}$$

we see that F' is a feasible aggregate flow for r. Equation (6) can then be rewritten

$$s + F' = F \tag{9}$$

Note that if OPT < 0, then there is a feasible solution to the LPP with $s > 0$, and from (9), F contains loops. Conversely, if OPT = 0, there is no $s > 0$ and feasible F' satisfying (9) and F is loopfree. Next note that the LPP always has at least one feasible solution, obtained by setting $s = 0$ and setting f' equal to the multicommodity flow corresponding to F. Since s is bounded, $0 \leq s \leq F$, the LPP must have an optimal feasible solution (although generally not unique). Finally, from (9) and (5), we see that the F' in any optimal feasible solution is loop free. Summarizing, we have the following theorem.

Theorem 1: Let F be a feasible aggregate flow for the input r. Then the minimum in (5) exists and the corresponding F' is loopfree; F is loopfree iff OPT = 0.

Next we derive the dual linear programming problem to that in (5) to (7). The standard form for a LPP is to choose $x \geq 0$ to minimize $c\cdot x$ subject to $A\cdot x = b$ for a given row vector c, matrix A, and column vector b. The dual LPP to this standard primal LPP (see, for example, Luenberger [2]) is to choose a row vector u to maximize $u\cdot b$ subject to $u\cdot A \leq c$. The duality theorem states that if both the primal and dual have feasible solutions, or if either has an optimal feasible solution, then both have optimal feasible solutions with the same value. In our case, since the primal always has an optimal feasible solution, the dual does also.

In order to visualize (5) to (7) in standard form, let x be made up of the L components of s followed by the $(n-1)L$ components of f', thus x has nL components. Choosing c as L components of value -1 followed by $(n-1)L$ zeros, (5) is equivalent to minimizing $c\cdot x$. It turns out that the dual variables will have more significance if we multiply (6) on both sides by -1. Then the vector b will be the components of -F followed by r.

Finally we must associate a dual variable with each equation in (6), (7). Let d_{ik} be associated with (6) for each $(i,k)\varepsilon\mathscr{L}$, let $D_i(j)$ be associated with (7) for each $i \neq j$, and let d, D be vectors with these components respectively. Letting u be the components of d followed by those of D, the dual problem is to form the maximum:

$$\text{OPT} = \max_{d,D} \; - \sum_{(i,k)\in\mathscr{L}} d_{ik}F_{ik} + \sum_{\substack{i,j \\ i\neq j}} D_i(j)r_i(j) \tag{10}$$

subject to $uA \leq c$. This constraint can be viewed as nL inequalities, one for each column of A, or equivalently, one for each component of x. Identifying the elements of A with the coefficients of s and f in (6) and (7), these nL inequalities can be written explicitly for each $(i,k)\in\mathscr{L}$ as

$$d_{ik} \geq 1 \tag{11}$$

$$D_i(j) \leq d_{ik} + D_k(j) \quad ; \quad \text{each } j \neq i,k \tag{12a}$$

$$D_i(k) \leq d_{ik} \tag{12b}$$

Inequality (10) corresponds to the variable s_{ik}, (12a) to the variables $f_{ik}(j)$, and (12b) to $f_{ik}(k)$. To avoid the notational inconvenience of constantly separating the case $j = k$ from $j \neq k$ in (12), we shall often use (12a) for both cases under the convention that $D_j(j)$ is a constant equal to zero.

In order to understand the meaning of (12), suppose that a path of links from node i to j consists of the links (i,k), (k,ℓ),...,(m,j). By applying (12) recursively to the final element in (12a), we get

$$D_i(j) \leq d_{ik} + d_{k\ell} + \dots + d_{mj} \tag{13}$$

If we interpret d_{ik} as a length associated with link (i,k), and take the distance on a path to be the sum of the link lengths on the path, then (13) states that $D_i(j)$ is less than or equal to the distance on each path from i to j for the given d. If, for a given d, we simply choose each $D_i(j)$ as the distance of the minimum distance path from i to j (according to the lengths d), then this D clearly satisfies (12). From (13), any other choice D' will satisfy D' < D. The quantity (10) to be maximized in the dual problem multiplies each $D_i(j)$ by the non-negative coefficient $r_i(j)$, establishing the following lemma.

Lemma 2: For given positive d, (10) is maximized over D, subject to (12) by setting $D_i(j)$, for each i,j, $i \neq j$, equal to the minimum distance (i.e., the length of the minimum distance path) from i to j for the given d.

Minimum distance paths are closely related to the concept of shortest route flows. Specifically, for a given set of positive link lengths d and the corresponding minimum distances D we say that a multicommodity flow f is a shortest route multicommodity flow for d if $f_{ik}(j) = 0$ whenever $D_i(j) < d_{ik}(j)$; i.e., whenever link (i,k) is not on a minimum distance path from i to j. Similarly we say that F is a shortest route aggregate flow for d if some multicommodity flow f corresponding to F is a shortest route multicommodity flow for d.

We now state the complementary slackness theorem of linear programming in the context of the primal and dual problems here and then show its relation to shortest route flows.

Theorem 2: Let s, f' be a feasible solution to the primal problem and d, D be a feasible solution to the dual. A necessary and sufficient set of conditions for both solutions to be optimal is for all $(i,k)\in\mathscr{L}$, $i \neq j$,

$$s_{ik} = 0 \text{ if } d_{ik} > 1 \quad ; \quad (i,k)\in\mathscr{L} \tag{14}$$

$$f'_{ik}(j) = 0 \text{ if } D_i(j) < d_{ik} + D_k(j) \tag{15}$$

The following theorem relates loop freedom to shortest route flows.

Theorem 3: Let F be a feasible aggregate flow. F is loopfree iff there is a set of positive link lengths d such that F is a shortest route aggregate flow for d. Furthermore, any such d, scaled up so that its components exceed one, yields an optimal solution to the dual problem, (10) to (12).

Proof: Let f be any multicommodity flow corresponding to F, and note that s = 0, f' = f yields a feasible solution to the primal problem. Assume first that F is loop free, so that the above solution is optimal. Choose link lengths $d_{ik} \geq 1$ and the associated minimum distances D to give an optimal solution to the dual. From Theorem 3, (15) is satisfied and hence f' = f is a shortest route flow. Next assume that f is a shortest route flow for some set of positive link lengths d'. Let $d_{ik} = d'_{ik}/\min_{(\ell,m)\in\mathscr{L}} d'_{\ell m}$ for all $(i,k)\in\mathscr{L}$. Thus $d_{ik} \geq 1$ and f is a shortest route flow for d. Since d, D is a feasible solution to the dual, and since (14) and (15) are satisfied, we have an optimal feasible solution to the primal and F is loopfree.

Part of the reason for our interest in this result stems from our conjecture that in good quasi-static routing algorithms, the act of changing a flow $f_{ik}(j)$ from zero to non-zero or vice versa (i.e. establishing or terminating routes) should be done much more slowly and carefully than the act of changing a flow between different positive values. Since the question of whether or not a multicommodity flow is a shortest route flow (for a given d) is simply a question of which flows are zero, it appears that these lengths should play a role in quasi-static algorithms.

Our next few results concern the basic optimal solutions of the primal and dual LPP's. The matrix A in these problems is an L + n(n-1) by Ln matrix with linearly independent rows. A matrix B is called a basis for the LPP if it consists of

$L + n(n-1)$ linearly independent columns of A (i.e., it is formed from A by deleting linearly dependent columns). An x satisfying $Ax = b$ is called a basic solution (with basis B) if the components of x corresponding to columns of A not in B are all zero. If we let x_B be the vector of components of x corresponding to the columns in B, then we see that x_B is uniquely determined by $Bx_0 = b$; thus there is one basic solution x corresponding to each basis B. If this basic x also satisfies $x \geq 0$, it is called a basic feasible solution. Another well known result of linear programming [2] (and the justification for the simplex algorithms) is that if (optimal) feasible solutions exist to the primal then a basic (optimal) feasible solution exists.

Basic dual solutions are defined in a similar way. Let c_B be the vector of components of c corresponding to the columns of A in B. Then u is a basic solution to the dual problem if it satisfies $uB = c_B$. Again u is unique, given B, and u is a basic feasible solution if it also satisfies $uA \leq c$.

Our problem now is to investigate what sets of columns of A form a basis B. The simplest approach is in terms of the dual constraints (11) and (12). We must find a subset of $L + n(n-1)$ of these inequalities, which when satisfied with equality, uniquely specify d, D; such a subset is called a set of basic equations, and of course makes up a basis B. First note that for each destination commodity j, (12) consists of a set of inequalities, one for each link not originating at node j. These are the only inequalities that involve the n-1 variables $D_i(j)$, $i \neq j$, and thus at least n-1 of them must be used as basic equations. Next observe that the basic equations from (12) for a given j must correspond to a set of links T_j that contain a spanning tree of the network (where the orientation of the links in the tree is immaterial). Otherwise there would be a nonempty set of nodes that were unconnected to j by the links of T_j, and the equations corresponding to T_j could not uniquely specify $D_i(j)$ for the i in the set. The equations corresponding to this spanning tree for a given destination j can now be rewritten to express each $D_i(j)$ as a sum of signed link distances from i to j, using the path in the tree from i to j. If the link (k,ℓ) on the path from i to j has the same direction as the path, then $+d_{k\ell}$ is used in the sum and otherwise $-d_{k\ell}$ is used.

We have now identified $n(n-1)$ of the basic equations that must be used to uniquely specify d, D (i.e. for each of n destinations, there are n-1 equations corresponding to the links of a spanning tree). These equations uniquely specify D as a function of d. Some destinations might have more than n-1 of their inequalities used as basic equations. Each additional such equation $D_i(j) = d_{ik} + D_k(j)$ can be written using our previous solution for $D_i(j)$ and $D_k(j)$; this equation now becomes the sum of signed link distances around a cycle set equal to zero. This is the same type of equation as one uses in an electrical network, setting the sum of the voltages around a cycle equal to zero. As is well known from electrical circuit theory (and from graph theory), there exist at most L - n+1 linearly independent such equations, and since these equations involve only d, this is a maximum over all destinations together. We conclude from this that at least n-1 of the inequalities $d_{ik} \geq 1$ must be included as equalities in any basis. Although we do not need it here, the links corresponding to these equalities must also include a spanning tree.

Finally, suppose we choose a basis with $n(n-1)$ spanning tree equations and exactly L - n+1 independent cycle equations. Since these cycle equations are homogeneous and form a basis for the set of cycle equations, the sum of the distances around every cycle must be zero; thus the dual solution from any such basis is not feasible. We summarize these results in the following theorem.

Theorem 4: Every basis for the LPP of (5) to (7) contains at least n-1 columns corresponding to slack variables s_{ik} and, for each j, there is a set T_j of at least n-1 links, which include a spanning tree, such that the column corresponding to $f'_{ik}(j)$ for each $(i,k) \in T_j$ is in the basis. If the basis is feasible for the dual, there are at least n columns corresponding to slack variables.

Next we consider the restrictions on a basis B if the corresponding primal basic solution is to be feasible. First assume that $r_i(j) > 0$ for all i,j, $i \neq j$. Then every feasible solution to the primal must contain, for each j, a directed spanning tree to j of links for which $f'_{ik}(j) > 0$. By a directed spanning tree to j we mean a tree such that each node i has a directed path to j through the tree links; i.e. a tree in which all links point toward j. Since an optimal basic feasible solution to the primal must have a basis for which the dual is also feasible, we see that the basic variables for such a basis include at least n slack variables, and for each j, a directed tree to j of variables $f'_{ik}(j)$. Because of the directed trees in the basis, it is easy to see that the corresponding optimal basic dual solution d, D has the property that the components of D are minimum distances corresponding to d.

Theorem 5: Let F be a feasible aggregate flow for input $r \geq 0$. In any optimum feasible solution to (5) to (7), F' (and F if F is loopfree) is a shortest route flow for some d such that (d,D) is a basic feasible solution to the dual and the components of D are minimum distances for d.

Proof: We have just seen that if $r_i(j) > 0$ for all i,j, $i \neq j$, then there is a basic optimal feasible solution (d,D) to the dual for which the components of D are minimum distances for d. From Theorem 2,

F' (and F if F is loopfree) is a shortest route flow for d in any optimal solution of the primal. If $r_i(j) = 0$ for one or more i,j, $i \neq j$, we let $r(\varepsilon)$ be an input formed by adding $\varepsilon > 0$ to each $r_i(j) = 0$. Similarly let $F_{ik}(\varepsilon) = F_{ik} + \varepsilon h_{ik}$ when h_{ik} for each $(i,k) \varepsilon \mathscr{L}$ is chosen to make $F(\varepsilon)$ a feasible aggregate flow for $r(\varepsilon)$. Let ε vary over a sequence $\varepsilon_i \to 0$. Since there are a finite number of basic feasible solutions to the dual, and these are independent of r,F, we see that one of these must satisfy Theorem 5 for an infinite set of ε_i, and thus must satisfy Theorem 5 in the limit $\varepsilon = 0$.

Theorem 4 and 5 have an important consequence with respect to the amount of alternate routing required to achieve a given F (or its loopfree reduction F'). Let $\alpha_i(j)$ be the number of links (i,k) for which $f'_{ik}(j) > 0$ for a given basic optimal solution to the primal, and let $\bar{\alpha}$ be the average of $\alpha_i(j)$ over the $n(n-1)$ node pairs i,j, $i \neq j$. Then $\bar{\alpha} = 1$ corresponds to no alternate routing. We have just shown that in addition to the directed spanning trees (which contribute 1 to each $\alpha_i(j)$), basic optimal solutions have at most $L - n$ additional non zero $f'_{ik}(j)$, over all i,k,j. This establishes the following theorem.

Theorem 6: The amount of alternate routing in a basic optimal solution to the LPP of (5) to (7) is bounded by

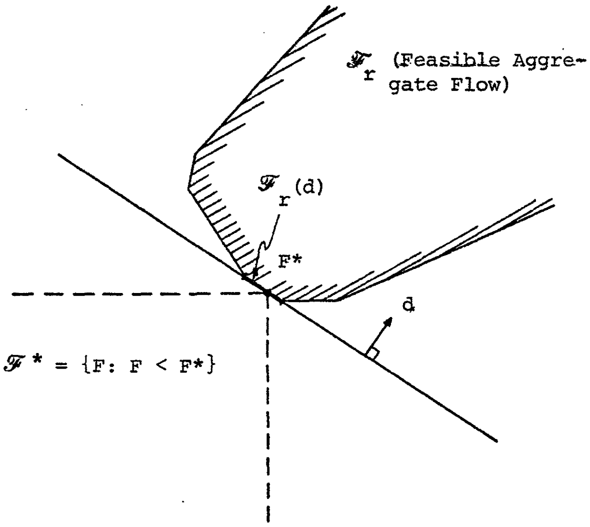
$$\bar{\alpha} \leq 1 + \frac{L-n}{n(n-1)} \tag{16}$$

Since L is typically much smaller than $n(n-1)$ in large networks, very little alternate routing is required to achieve any desired loopfree aggregate flow.

In order to obtain a better understanding of the meaning of Theorem 5, let $\mathscr{F}_r$ be the set of all feasible aggregate flows F for a given input r, and let $\mathscr{F}_r(d)$ be the set of shortest route aggregate flows for input r and link length assignment d. Let $\mathscr{B}$ be the set of link length assignments d for which (d,D) is a basic feasible solution to (10) to (12) and for which D is the set of minimum distances for d. Theorem 5 then asserts that the set of loopfree flows for r, LFF_r, as given by

$$LFF_r = \bigcup_{d \varepsilon \mathscr{B}} \mathscr{F}_r(d) \tag{17}$$

We can interpret $\mathscr{F}_r(d)$ (for any d with positive components) as the set of vectors F that minimumize $F \cdot d$ over the constraint that F be a feasible aggregate flow for r (this comes directly from the definition of a shortest route aggregate flow). We also obtain the geometric interpretation from Figure 2 that if F minimizes $F \cdot d$ over $F \varepsilon \mathscr{F}_r$, than any vector F' for which $F' < F$ must satisfy $F' \cdot d < F \cdot d$; thus $F' < F$ implies $F' \notin \mathscr{F}_r$, so that



Geometric Interpretation

Figure 2

F is loopfree (a fact we already know from Theorem 3). More compactly, this says that $\mathscr{F}_r(d)$ is a face of $\mathscr{F}_r$, i.e. the subset of $\mathscr{F}_r$ that lies on the supporting hyperplane of $\mathscr{F}_r$ with normal vector d.

Theorem 7: Let d be an arbitrary positive length vector. Then $\mathscr{F}_r(d) \subset \mathscr{F}_r(d')$ for some $d' \varepsilon \mathscr{B}$.

We postpone the proof of this until after the proof of Theorem 8. The theorem says that all the shortest route flows for any given positive d are included in the shortest route flows for one of the basic distance vectors in $\mathscr{B}$. Geometrically, it says that the faces $\mathscr{F}_r(d)$ for $d \not\varepsilon \mathscr{B}$ are in some sense superfluous, since each of them is contained in some basic face $\mathscr{F}_r(d')$. It should also be noted from the dual problem (10) to (12) that the set of basic distance vectors is independent of the input r. Thus as r changes, each of the basic faces $\mathscr{F}_r(d)$ move in space but maintain the same normal.

Some additional interpretation of these faces arises from defining

$$\mathscr{R}(d) = \{(i,k,j): D_i(j) = d_{ik} + D_k(j)\} \tag{18}$$

where D is the set of minimum distances for d. $\mathscr{F}_r(d)$, then, is the set of aggregate flows for which a corresponding multicommodity flow exists in which each commodity j flows only over links (i,k) for which (i,k,j) is in $\mathscr{R}(d)$. This means that $\mathscr{F}_r(d)$ is a function of d only through the set $\mathscr{R}(d)$. It turns out that several basic distance vectors can have the same $\mathscr{R}(d)$ (in which case they determine the same face $\mathscr{F}_r(d)$). It can also happen that $\mathscr{R}(d)$ for one basic d can be strictly contained in $\mathscr{R}(d')$ for another basic d', in which case

$\mathscr{F}_r(d)$ is strictly contained in $\mathscr{F}_r(d')$ (assuming positive r). Thus, in some sense, a more elegent representation of the class of loop free flows could be obtained directly in terms of a minimal class of sets $\mathscr{R}(d)$, but we have been unable to do this except through the basic distance vectors.

Theorem 8: Let $F_1, F_2, \ldots, F_m$ be feasible aggregate flows for input r and let $\lambda_1, \lambda_2, \ldots, \lambda_m$ be positive numbers summing to 1. Then $F_0 = \sum_{i=1}^{m} \lambda_i F_i$ is in $\mathscr{F}_r(d)$ iff $F_i \in \mathscr{F}_r(d)$ for $1 \leq i \leq m$.

Discussion: This says that the convex combination of loop free flows is not in general loopfree unless each of the combined flows are shortest route flows for the same d. This theorem appears to give an indication why routing algorithms such as the Cantor-Gerla algorithm [3], which operate by taking convex combinations of extremal flows (i.e. flows that are extremal points of the set $\mathscr{F}_r$), converge quickly at first and then more slowly. Each new extremal flow, when combined with the others, typically adds a new set of loops to the solution, and these loops are eliminated only when all of the extremal flows being used are shortest route for the same d.

Proof of Theorem 8: Let $c = \min F \cdot d$ over $F \in \mathscr{F}_r$.

$$F_0 \cdot d = \sum_{i=1}^{m} \lambda_i F_i \cdot d \geq c$$

with equality iff $F_i \cdot d = c$ for $1 \leq i \leq m$.

Proof of Theorem 7: Suppose, to the contrary that $\mathscr{F}_r(d)$ is not contained in $\mathscr{F}_r(d')$ for any $d' \in \mathscr{B}$. Let $d_1, d_2, \ldots, d_m$ be the elements of $\mathscr{B}$ and let F_i be an aggregate flow in $\mathscr{F}_r(d)$ but not in $\mathscr{F}_r(d_i)$; the F_i need not be distinct. Let $\lambda_1, \ldots, \lambda_m$ be positive numbers summing to 1, and let $F_0 = \sum_{i=1}^{m} \lambda_i F_i$. From Theorem 8, F_0 is not in $\mathscr{F}_r(d_i)$, for $1 \leq i \leq m$, and from (17) F_0 is not a loopfree flow. However, again from Theorem 8, F_0 is in $\mathscr{F}_r(d)$, and hence is a loopfree flow, supplying the desired contradiction.

Acknowledgement

The author is grateful to D. Bertsekis, A. O'Leary, and J. Wozencraft for many helpful discussions about this topic.

References

1. R. Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation", IEEE Trans. Commun., Vol. COM-25, No. 1, Jan. 1977, pp. 73-85.

2. D. G. Luenberger, Introduction to Linear and Non-Linear Programming, Reading, Ma., Addison Wesley, 1973.

3. D. G. Cantor and M. Gerla, "Optimal Routing in a Packet Switched Computer Network", IEEE Trans. Comput., Vol. C-23, pp. 1062-1069, Oct. 1974.