Lexicographically Optimal Balanced Networks

Leonidas Georgiadis

Aristotle Univ. of Thessaloniki

Dept. of Elect. and Comp. Eng

P.O. Box 435

Thessaloniki 54006, GREECE

e-mail: leonid@eng.auth.gr

Panos Georgatsos

A.T.T.S. Dept, Algosystems S.A.

4 Sardeon str, 171 21 N.Smyrni

Athens, GREECE

e-mail: pgeorgat@algo.com.gr

Kostas Floros

Aristotle Univ. of Thessaloniki

Dept. of Elect. and Comp. Eng

P.O. Box 435

Thessaloniki 54006, GREECE

e-mail: kflo@intranet.gr

Stelios Sartzetakis

ICS-FORTH

Science and Technology Park of Crete

P.O. Box 1385, Heraklion-CRETE

GREECE GR-711 10

e-mail: stelios@ics.forth.gr

Abstract— **We consider the problem of allocating bandwidth between two endpoints of a backbone network so that no parts of the network are unnecessarily loaded. We formulate the problem as lexicographic optimization, and develop algorithms for its solution. The solution consists of a) identifying a cut in the network where the optimal load can be determined on all the links of the cut, and b) considering the same problem for each of the subnetworks to which the cut is dividing the original network.**

I. Introduction

Consider the situation depicted in Figure 1. Private network A needs a total bandwidth B_w in order to satisfy its communication needs with private network B. This is a practical situation arising to any connectivity provider domain, offering (semi-permanent) connection services to its customers (e.g. ISP providing VPN or leased line services). In the above context, the treatment of (semi-permanent) connection requests so that the provider's domain remains in a "balanced" state, is the main concern of the paper. Of course, there are many other issues that need to be addressed in this situation, such as, network design, resource dimensioning, reliability , routing and admission control techniques, assurance that the selected paths satisfy quality of service related constraints etc. [16], [9], [10], [3], [19]. However, within our framework, the allocation of bandwidth in a manner that avoids overloading parts of the network is a basic concern and can serve as building block for addressing problems related to the above mentioned issues. Usually bandwidth allocation and the accompanying route determination is related to an optimization problem with objective function representing an overall network cost, or performance measure such as average delay, or average loss probability [17], [8], [10], [19]. The model we consider in

this paper is different. It is assumed that the customer requests from the network certain bandwidth. This bandwidth is to be used by various customer applications, and is sufficient to satisfy the customer's desired call blocking probability. In addition, if the notion of equivalent bandwidth is incorporated into the model [13], [11], [15], [7], then certain QoS criteria per customer, e.g., loss probability can be taken into account. The network then seeks to provide the requested bandwidth without overloading unnecessarily any part of the network. As a result the network

• may be able to accept unpredictable traffic reservation requests that may occur in various parts of the network
• can accommodate best effort traffic that may be generated in an unpredictable manner in various parts of the network
• facilitates the process of finding alternate routes in case of component failures [20].

The bandwidth allocation thus obtained can be used as a basis for dynamic resource management. For example, the routes obtained from such a bandwidth allocation may be used as candidate routes for performing dynamic admission control [14], [12], [18]: when a new connection request arrives, the candidate routes are searched (the route bandwidth can affect the order of search) and if there are enough resources in one of routes, the connection is admitted. Hence, overloading parts of the network is avoided by design. In addition, since only a small number of routes needs to be monitored, the complexity of the admission control is reduced.

The problem we consider in this paper is how to pick the routes and the associated route bandwidth within the backbone network, so that the bandwidth request is satisfied and the backbone network is left in a "balanced" state. Intuitively a network is balanced if the bandwidth allocation results in an even spreading of the load to various links of the network. The simplest optimization criterion that reflects this requirement is the minimization of the load
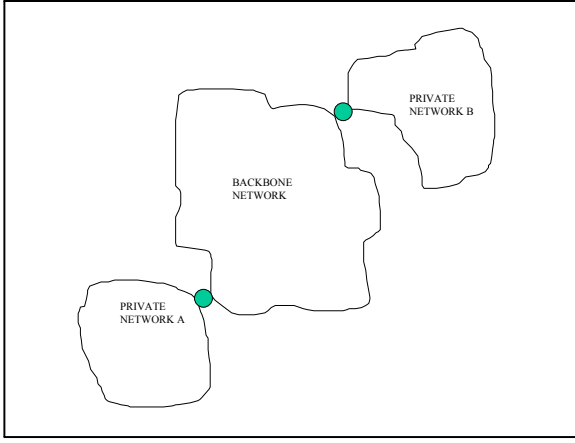
Fig. 1. System Configuration

Fig. 2. Methods of Bandwidth Allocation

of the maximally loaded link in the network. This criterion is one of the basic requirements from Internet Service Providers (ISPs) [4], [21]. However, such an optimization does not provide a means of allocating bandwidth to links that are not maximally loaded. In this paper, we consider a stronger minimization criterion, namely, lexicographic optimization. More specifically, we consider a network as balanced if the vector of bandwidths allocated to each link is lexicographically optimal with respect to certain link costs representing the degree by which the link is loaded (also known as a min-max vector [6, page 526]). Lexicographic optimization attempts to first reduce as much as possible the load of the maximally loaded link in the network. Next, if there are many choices available, it attempts to reduce as much as possible the second maximally loaded link cost in the network, and so on. With the appropriate choice of link cost functions, the degree of loading of a link may be adapted. Hence certain links may be kept more or less loaded according to administrative policies.

An example of the effect of balancing the network load in a lexicographically optimal fashion is given in figure 2. Assume that all links have the same capacity equal to one, and that one unit of bandwidth is to be allocated. Assume also that the link utilization is taken as the link cost. In part a), this bandwidth is allocated on a single path, with the result that all links on the path have load 1. In part b), the allocation is made over two separate paths, and as result the load on the six links of these paths is 1/2. This is one of the allocations that could result if one is trying to minimize the maximum loaded link in the network. In part c), the lexicographically optimal allocation is shown. In this case, in effect four paths are chosen and the resulting link loads are 1/2 on two of the links and 1/4 on the rest. Hence there are four more links with maximum load in case b) than in case c), that is, case c) leaves the network in a more balanced state.

Using lexicographic optimization of link cost functions as the criterion for load balancing, we develop in this paper algorithms for providing the optimal solution. Link cost are continuous and increasing functions of link loads and may be so defined to incorporate varying link capacities. The
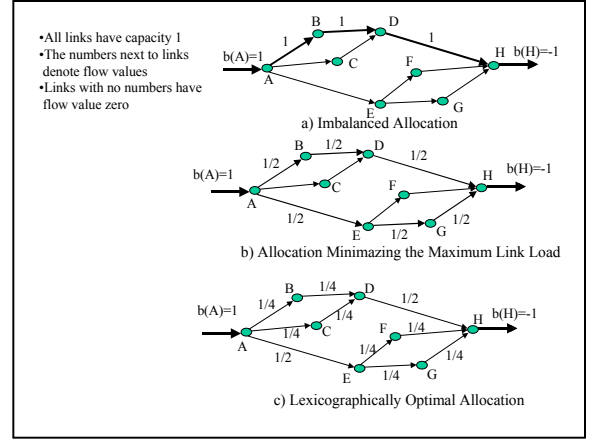
solution is recursive and consists of a) identifying a cut in the network where the optimal link load can be determined on all links of the cut, and b) considering the same problem for each of the subnetworks to which the cut is dividing the original network. For general link cost functions the solution may take exponential number of steps, while for piecewise linear cost functions the required number of steps is polynomial in the network size and the maximal number of linear pieces in any link cost function.

The rest of the paper is organized as follows. In Section II we present the notation used in the paper and some prior results. In Section III we formulate the optimization problems of interest and provide the basic approach to the solution. Numerical results are presented in Section IV. We summarize our results and provide suggestions for further work in Section V. Finally, in the Appendix we provide the proofs of Lemmas used in the paper.

II. Notation Definitions and Prior Results

Given an n-dimensional real vector x define by $\phi(x)$ the n-dimensional vector whose coordinates are those of x arranged in non-increasing order, i.e.,

$$\phi(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_n(x)) = (x_{i_1}, x_{i_2}, \ldots, x_{i_n})$$

where $x_{i_1} \geq x_{i_2} \geq \ldots \geq x_{i_n}$. Vector x is called *lexicographically smaller* than or equal to vector y, if either $\phi(x) = \phi(y)$, or there exists a number l, $1 \leq l \leq n$ such that $\phi_i(x) = \phi_i(y)$, for $1 \leq i \leq l-1$ and $\phi_l(x) < \phi_l(y)$. We write $x \preceq y$, and if in addition $\phi(x) \neq \phi(y)$, $x \prec y$.

In the example of Figure 2, the vector of link loads in case c) is lexicographically smaller than the vector of link loads in case b), which in turn is lexicographically smaller than the vector of link loads in case a).

The following observation, which is immediate from the definition, will be needed in the sequel.

Property 1. *Let x and y be n-dimensional vectors such that $x_i = y_i$ for all i in a set $M \subseteq \{1, \ldots, n\}$ and define $\overline{M} = \{1, \ldots, n\} - M$. If $\max_{i \in \overline{M}}\{x_i\} < \max_{i \in \overline{M}}\{y_i\}$, then $x \prec y$.*

In the sequel we will need the following definitions and notations. Consider a network $G = (N, A)$ with a set N of nodes and a set A of directed arcs (links).

Let a quantity be indexed by a link $l \in A$, say W_l, where $l = (i, j)$, $i, j \in N$. To avoid cumbersome notation, instead of $W_{(i,j)}$ we write W_{ij}.

A *cut* is a set of links whose removal divides network G in two disconnected subnetworks $G_1 = (S, A_1)$, and $G_2 = (\overline{S}, A_1)$, where $\overline{S} = N - S$. The set of *forward* links, i.e., links with tail in set S and head in set $\overline{S}$ is denoted by $(S, \overline{S})$. Similarly, the set of *backward* links, i.e., links with head in set S and tail in set $\overline{S}$ is denoted by $(\overline{S}, S)$. We also denote the set of links in the cut as, $[S, \overline{S}] = (S, \overline{S}) \cup (\overline{S}, S)$.

The *network induced* by node set $S \subseteq N$, is a network with node set S and link set the links in A with both endpoints in the set S.

Let the real numbers $b(i)$, $u_{ij} \geq 0$, be given for any $i \in N$ and $(i, j) \in A$. The number u_{ij} will be referred to as the "capacity" of link (i, j). Assume also that $\sum_{i \in N} b(i) = 0$. We denote $N_s = \{i \in N : b(i) > 0\}$ and $N_t = \{i \in N : b(i) < 0\}$. For a set $S \subseteq N$ we define $b(S) = \sum_{i \in S} b(i)$

The capacity of a cut $[S, \overline{S}]$ is defined as the sum of the capacities of the forward links of the cut, i.e.,

$$U[S, \overline{S}] = \sum_{i \in (S, \overline{S})} u_{ij}.$$

Note: In the discussion below, u_{ij} will not necessarily represent the physical link capacities in bps. To avoid confusion, we will refer to the latter explicitly as the "physical link capacity".

The following theorem will be needed [2, Theorem 6.12 , page 196] in the sequel.

Theorem 1: The system of constraints

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b(i), \text{ for all } i \in N \qquad (1)$$

$$0 \leq x_{ij} \leq u_{ij}, \text{ for all } (i, j) \in A \qquad (2)$$

has a feasible solution if and only if for every subset $S \subseteq N$, $b(S) - U[S, \overline{S}] \leq 0$.

A vector $x = \{x_{ij}\}_{(i,j) \in A}$ that satisfies the constraints of Theorem 1 and in addition does not contain positive cycles (i.e., cycles $(i_1, ..., i_K = i_1)$ such that $\min_k \{x_{i_{k-1} i_k}\} > 0$) will be referred to as a "flow" on G. We denote by $F_{G,b,u}$ the set of all flows on G. Whenever there is no possibility for confusion we use the simpler notation F_G.

The problem of finding a feasible flow satisfying the constraints (1) and (2) can be solved by solving a maximum flow problem defined on an augmented network [2]. Specifically, network G is augmented with two nodes s, t, and with arcs (s, i) for $i \in N_s$, and (i, t) for $i \in N_t$. The capacity of an arc (s, i) for $i \in N_s$ is $b(i)$ and the capacity of an arc (i, t) for $i \in N_t$ is $-b(i)$. Then the maximum flow from s to t in the augmented network is solved. If the maximum flow saturates all links (s, i), $i \in N_s$ and (i, t), $i \in N_t$, then there is a feasible flow satisfying the constraints (1) and (2). Otherwise, there is no feasible solution.

III. Problem Formulation and Solution

We will need to address a more general problem than the one presented in the introduction, which is also of interest. Consider a network $G = (N, A)$. Assume that there is a set N_s of supplier (ingress access) nodes and a set N_t of consumer (egress access) nodes, with $N_s \cap N_t = \emptyset$. A node $i \in N_s$ produces $b(i) > 0$ units of a commodity (bandwidth) and a node $i \in N_t$ consumes $b(i) < 0$ units of the same commodity. For the rest of the nodes (transit nodes), $N - N_s - N_t$ assume that $b(i) = 0$. To avoid trivial cases we assume that $N_s \neq \emptyset$. We also assume that

$$\sum_{i \in N} b(i) = 0.$$

For the problem discussed in the introduction, we have $|N_s| = |N_t| = 1$, i.e., there is one supplier node (ingress node of private network A to backbone network) and one consumer node (egress node of backbone network to private network B); nodes with zero $b(i)$ are transit nodes of the backbone network.

Assume that physical link capacities for network G are infinite (see the note below on how to incorporate physical link capacities into the problem). Let x be a flow on G. With each link $l \in A$, there is an associated cost $C_l(x_l)$, where $C_l(x_l)$ is a continuous strictly increasing function. We assume that $C_l(0) = 0$ for all $l \in A$. The problem we consider is to find a flow $x = \{x_l\}_{l \in A}$ such that the induced vector of link costs $\{C_l(x_l)\}_{l \in A}$ is lexicographically optimal:

Problem A: *Among the vectors $x = \{x_l\}_{l \in A}$ satisfying the constraints*

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b(i) \text{ for all } i \in N$$

$$x_{ij} \geq 0 \text{ for all } (i, j) \in A$$

find one, $\widehat{x}$, such that the vector of link costs $\{C_l(\widehat{x}_l)\}_{l \in A}$ is lexicographically optimal.

A solution to problem A will be referred to as a "lexicographically optimal" flow.

The finding of a lexicographically optimal flow will be based on the properties of the solution to the following min-max problem.

Problem B: *Solve*

$$\min_{x \in F_G} \max_{l \in A} \{C_l(x_l)\},$$

where

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = b(i) \text{ for all } i \in N$$

$$x_{ij} \geq 0 \text{ for all } (i, j) \in A$$

In the following we assume that the constraint set of problems A and B is not empty. This will be the case if, for

example, there is a path in G from any node in N_s to any node in N_t.

Note: The physical link capacities are not included in the constraints of problem A or B. These constraints can be taken into consideration by defining appropriately the cost functions. Specifically, let B_l be the physical capacity of link l. Let the cost functions satisfy the condition

$$C_l(B_l) = U > 0, \text{ for any link } l.$$

Then, if there is a flow x satisfying the physical link capacity constraints, the solution to problem A or B will also satisfy these constraints. Indeed, since $x_l \leq B_l$ and the cost functions are increasing, we have

$$\max_{l \in A} \{C_l(x_l)\} \leq \max_{l \in A} \{C_l(B_l)\} = U.$$

On the other hand, for an optimal flow $\widehat{x}$ of problem A or B, we have for any link k,

$$C_k(\widehat{x}_k) \leq \max_{l \in A} \{C_l(\widehat{x}_l)\} \leq \max_{l \in A} \{C_l(x_l)\} \leq U = C_k(B_k),$$

and therefore, $\widehat{x}_k \leq B_k$. Hence, with this choice of cost functions, if it turns out that the solution to problems A or B does not satisfy the physical link capacity constraints, then we know that no feasible solution exists for the problem.

Note that without loss of generality we may restrict attention to flows that satisfy

$$x_l \leq \sum_{i \in N_s} b(i).$$

Thus the constraint set is compact, and since we also assume that it is nonempty, there exists a solution to problem B. Using the fact that the set of solutions to problem B is compact, it can also be shown that a solution to problem A exists (see the Appendix for an outline of the argument). We will refer to these solutions in the sequel without further comments.

Given a number $\alpha \geq 0$, define the capacity of link l as $u_l(\alpha) = C_l^{-1}(\alpha)$, where $C_l^{-1}(\cdot)$ is the inverse of $C_l(\cdot)$. Let also $U_\alpha[S, \overline{S}]$ be the capacity of the cut $[S, \overline{S}]$:

$$U_\alpha[S, \overline{S}] = \sum_{(i,j) \in (S\overline{S})} u_{ij}(\alpha).$$

The following two lemmas provide the basic properties of a solution to problem B, on which the solution to problem A will be based. The proofs of all lemmas, corollaries and theorems are given in the appendix.

Lemma 2: Let x^* solve Problem B and let $\beta = \max_{l \in A} \{C_l(x_l^*)\} > 0$. Then there is a node set $T \subseteq N$, such that $b(T) = U_\beta[T, \overline{T}] > 0$.

Lemma 3: a) Let x^* solve Problem B. Let the node set $T \subseteq N$, be such that $b(T) = U_\beta[T, \overline{T}] > 0$. Any flow x' that solves Problem B satisfies

$$C_l(x_l') = \beta, \text{for every } l \in (T, \overline{T}) \tag{3}$$
$$C_l(x_l') = 0, \text{for every } l \in (\overline{T}, T) \tag{4}$$

where $\beta = \max_{l \in A} \{C_l(x_l^*)\} > 0$.

b)Conversely, if a flow $x' \in F_{G,b,u(\beta)}$ satisfies conditions (3) and (4) for some set T with $(T, \overline{T}) \neq \emptyset$ and some number $\beta > 0$, then $b(T) = U_\beta[T, \overline{T}] > 0$, x' solves problem B and the optimal value is β.

A link k such that $C_k(x_k) = \beta = \max_{l \in A} \{C_l(x_l)\}$ will be called "maximally loaded". A cut $[T, \overline{T}]$ such that $b(T) = U_\beta[T, \overline{T}] > 0$ will also be called maximally loaded.

The previous two lemmas state that for any solution to problem B there is a set T such that a) all links emanating from T are maximally loaded and b) there is no load on links entering T. Moreover, these two properties of set T remain true under any other solution to problem B. Since the cost functions $C_l(x)$ are strictly increasing, this implies that all solutions to problem B have the same values on the links of the cut $[T, \overline{T}]$. Consider for example the solution to problem B in Figure 2 b). The set T in this case is $T = \{A, B, C, D\}$, and the corresponding cut is $[T, \overline{T}] = \{(A, E), (D, H)\}$. Hence $x_{(A,B)} = x_{(D,H)} = 1/2$ for all solutions to problem B. Note that the same is not true for other maximally loaded links, e.g., link (B, D).

Assume for the moment that β and a maximally loaded cut $[T, \overline{T}]$ are known, and let $\widehat{x}$ be a lexicographically optimal flow. Since $\widehat{x}$ solves problem B, according to the previous argument, the values of $\widehat{x}$ on any link in $(\overline{T}, T)$ are zero, and the value on any link l in $(T, \overline{T})$ is $\widehat{x}_l = C_l^{-1}(\beta) > 0$. Therefore, it remains to find the values of the lexicographically optimal flow on the links of each of the networks induced by the nodes in the sets T and $\overline{T}$. Let us denote these networks by $G_T = (T, A_T)$ and $G_{\overline{T}} = (\overline{T}, A_{\overline{T}})$ respectively. Define next for any node i in network G_T,

$$b_T(i) = b(i) - \sum_{j:(i,j) \in (T,\overline{T})} C_{ij}^{-1}(\beta) \tag{5}$$

(we use the convention that summation over an empty set is zero). Similarly, define for any node i in the network $G_{\overline{T}}$,

$$b_{\overline{T}}(i) = b(i) + \sum_{j:(j,i) \in (T,\overline{T})} C_{ij}^{-1}(\beta). \tag{6}$$

Within each of the networks G_T and $G_{\overline{T}}$, $\widehat{x}$ solves problem B with the parameters b_T, $b_{\overline{T}}$ respectively. Indeed, assume that $\widehat{x}$ does not solve problem B in, say, network G_T. Then there is a flow y on G_T such that

$$\max_{(i,j) \in A_T} \{C_{ij}(y_{ij})\} < \max_{(i,j) \in A_T} \{C_{ij}(\widehat{x}_{ij})\}.$$

Consider the vector $\{z_{ij}\}_{(i,j) \in A}$ defined as follows.

$$z_{ij} = \begin{cases} \widehat{x}_{ij} & \text{if } (i,j) \notin A_T \\ y_{ij} & \text{if } (i,j) \in A_T \end{cases}$$

Such a definition constitutes a feasible flow on G. Moreover, by Property 1, we have that $z \prec \widehat{x}$, which contradicts the assumption that $\widehat{x}$ is optimal.

To summarize, we have seen that starting from problem B defined on network G, and provided that we know the optimal value β and a maximally loaded cut $[T, \overline{T}]$,

- we determine uniquely the lexicographically optimal values of the flow on the links of the cut
- to find the lexicographically optimal flow on the rest of the links, we need to solve problem B on the smaller (in terms of links and nodes) networks G_T and $G_{\overline{T}}$

Continuing the subdivision of the networks in this fashion, we eventually end up either with a single node network, or with a network for which $b_T(i) = 0$ for all $i \in A_T$, for which the solution is obviously $x_l = 0$ for all links. Therefore, we eventually determine the values of the lexicographically optimal flow on all the links. Note that since the values of the lexicographically optimal flows are uniquely determined on each of the maximally loaded cuts, the resulting lexicographically optimal flow is also unique. We state this fact as a corollary.

Corollary 4: The lexicographically optimal flow (solution to Problem A) is unique.

The basic algorithm described above, is shown in the following pseudocode.

algorithm A:
INPUT: A network $G = (N, A)$, link cost functions $C_l(x)$ for all $l \in A$, node commodities $b(i)$, for all $i \in N$.
OUTPUT: Lexicographically optimal flow $\widehat{x}$.
lexicographically_optimal_flow(G, C, b):
  if $|N| = 1$ return;
  if $(b(i) = 0$ for all $i \in N)$ then $\{\widehat{x}_l = 0$ for all $l \in A$; return($\widehat{x}_l$);$\}$
  $\beta =$ min_max(G, C_l, b).value;
  $x^* =$ min_max(G, C, b).argument;
  $[T, \overline{T}] =$ maximally_loaded_cut(G, C, b);
  for $\{$all $l \in [T, \overline{T}]\}$ do $\widehat{x}_l = x_l^*$; end;
  lexicographically_optimal_flow($G_T, \mathsf{C}^\mathsf{T}, \mathsf{b_T}$); /*$C_l^T(x) = C_l(x)$ for all $l \in T$ */
  lexicographically_optimal_flow($G_{\overline{T}}, C^{\overline{T}}, b_{\overline{T}}$); /*$C_l^{\overline{T}}(x) = C_l(x)$ for all $l \in \overline{T}$ */
return($\widehat{x}$);

It is important to note that there may be more than one maximally loaded cuts in the network. Therefore, the speed of the algorithm will increase if we identify the links that belong to all such cuts and then set their values accordingly. Consider for example the network in Figure 3. One maximally loaded cut is $\{(L, E), (D, H)\}$. Hence the two smaller networks on which Problem B will be solved next are those induced by the node sets $T = \{E, F, G, H\}$ and $\overline{T} = \{I, K, J, L, A, B, C, D\}$. However link (L, A) also belongs to a maximally loaded cut, i.e. the cut $\{(L, E), (L, A)\}$. Hence the lexicographically optimal flow on link (L, A) is also .5 and we can proceed by solving Problem B on the networks induced by the node sets $T = \{E, F, G, H\}$, $T_1 = \{A, B, C, D\}$ and $T_2 = \{I, J, K, L\}$.

Next we propose a method for finding all the links that belong to the maximally loaded cuts in network G if a solution to problem B, x^*, is known. Define a network $\widetilde{G}(N, \widetilde{A})$, where
- The direction of a link such that $x_l^* = 0$ (zero loaded links) is inverted
- A link that is neither maximally loaded nor zero loaded is replaced with a bidirectional link (or two links, one in
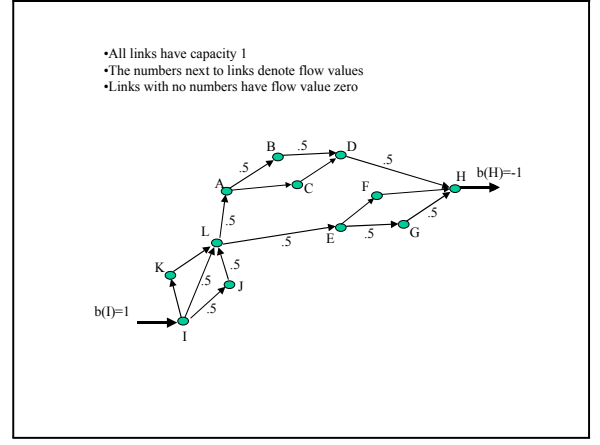

Fig. 3. Multiple Maximally Loaded Cuts.

each direction)

Note that since $C_l(0) = 0$, and the cost functions are strictly increasing, it is not possible for a node to simultaneously be maximally and zero loaded (barring the trivial case where $b(i) = 0$ for all $i \in N$).

If $[T, \overline{T}]$ is a maximally loaded cut, then there is no path in $\widetilde{G}(N, \widetilde{A})$ from a node in $\overline{T}$ to a node in T. Therefore network $\widetilde{G}(N, \widetilde{A})$ has at least two strongly connected components. Assume now that we find the strongly connected components of $\widetilde{G}(N, \widetilde{A})$, $\widetilde{G}_1, \widetilde{G}_2, \ldots, \widetilde{G}_L, L \geq 2$. Observe that any maximally loaded cut $[T, \overline{T}]$ partitions the strongly connected components into two subsets such that

$$T = \cup_{i \in s_1} \widetilde{G}_i, \overline{T} = \cup_{i \in s_2} \widetilde{G}_i, s_1 \cap s_2 = \emptyset, \; s_1 \cup s_2 = \{1, .., L\}$$

Therefore, the links that belong to maximally loaded cuts are links that connect the components $\widetilde{G}_1, \widetilde{G}_2, \ldots, \widetilde{G}_L$. In fact, the load of all the links that connect these components is the load induced by the lexicographically optimal point. This fact is based on the following lemma.

Lemma 5: Any link that connects components $\widetilde{G}_1, \widetilde{G}_2, \ldots, \widetilde{G}_L$ has either zero load under any flow, or belongs to one or the maximally loaded cuts of network G.

Combining now the previous lemma with Lemma 3 a) we see that the load of the links that connect the components $\widetilde{G}_1, \widetilde{G}_2, \ldots, \widetilde{G}_L$ is the load induced by the lexicographically optimal point.

The modified algorithm is given in the pseudocode that follows. In this pseudocode, G_k is the subnetwork of network G having the same nodes as component $\widetilde{G}_k$ of $\widetilde{G}$, and b_k are the respective commodities, determined in a manner similar to (5), (6).

algorithm B:
INPUT: A network $G = (N, A)$, link cost functions $C_l(x)$ for all $l \in A$, node commodities $b(i)$, for all $i \in N$.
OUTPUT: Lexicographically optimal link load vector $\widehat{x}$.
lexicographically_optimal_flow(G, C, b):
  If $|N| = 1$ return;
  If $(b(i) = 0$ for all $i \in N)$ then $\{\widehat{x}_l = 0$ for all $l \in A$; return($\widehat{x}_l$);$\}$
  $\beta =$ min_max(G, C, b).value;

```
x* =min_max(G, C, b).argument;
G̃ =create_network(G, x*) ;
{G̃₁, G̃₂, ..., G̃_L} =strongly_connected_components(G̃)
  for {all links connecting the strongly connected compo-
nents{G̃₁, G̃₂, ..., G̃_L}}
    do x̂_l = x*_l; end;
  for k = 1 to L
    do lexicographically_optimal_flow(G_k, b_k);end;
return(x̂);
```

Procedure create_network
INPUT: Network $G(\overline{N}, A)$, x flow on G.
OUTPUT: A network consisting of the same nodes as G and
with modified links.
```
create_network(G, x)
  Ã = A;
  for l ∈ A do
    if (x_l = 0) then invert direction of link l in Ã;
    else if ( l is not maximally loaded) replace link  l in Ã
with a bidirectional link;
  end;
return G̃(N, Ã);
```

It remains to determine a method for finding β and x^*.

This is presented in the next section.

A. Algorithm for Solving Problem B

In [1], an algorithm for solving Problem B for the trans-
portation problem and for linear cost functions was pre-
sented. The general approach in [1] can be adapted to
the problem at hand. However, intricacies arise due to the
fact that we consider general networks and nonlinear cost
functions. We address these issues in this section.

We assume that the solution in α to the equation

$$\sum_{l \in S} C_l^{-1}(\alpha) = r, \; S \subset A, \; r \geq 0 \tag{7}$$

can be computed exactly (or with very high accuracy), as,
for example, in the case where the costs are piecewise linear
functions.

First, a positive lower bound on the value z^* of Problem
B is determined based on the following lemma.

Lemma 6: Let the cut $[T_0, \overline{T}_0]$ be such that $N_s \subseteq T_0$. If
$z_0 > 0$ is the solution to equation

$$\sum_{l \in (T_0, \overline{T}_0)} C_l^{-1}(z_0) = \sum_{i \in N_s} b(i), \tag{8}$$

then, $z_0 \leq z^*$.

Note that we can pick $T_0 = N_s$ and hence obtain a lower
bound z_0 according to Lemma 6. Having determined z_0
we can proceed to compute an increasing sequence of lower
bounds as follows. Let $z_k \leq z^*$ be a lower bound. Augment
network G with two nodes s, t, and with arcs (s, i), $i \in N_s$,
(i, t), $i \in N_t$. Also, define the link capacities, $u_l = C_l^{-1}(z_k)$,
$l \in A$ and $u_{(s,i)} = b(i)$, $i \in N_s$, $u_{(i,t)} = -b(i)$, $i \in N_t$. Let

$\widehat{G}_k$ be the augmented network. The next lemma provides
a means of improving the lower bound.

Lemma 7: Let $x^{(k)}$ be the maximum flow on the aug-
mented network $\widehat{G}_k$. If $x^{(k)}$ saturates all links (s, i),
$i \in N_s$, then $x^{(k)}$ solves Problem B. Otherwise, any cut
$[T_{k+1}, \overline{T}_{k+1}]$ in G such that $\left[\{s\} \cup T_{k+1}, \overline{\{s\} \cup T_{k+1}}\right]$ is a
minimum cut in $\widehat{G}_k$ satisfies

$$\sum_{i \in T_{k+1}} b(i) > \sum_{l \in (T_{k+1}, \overline{T}_{k+1})} C_l^{-1}(z_k).$$

If z_{k+1} solves

$$\sum_{i \in T_{k+1}} b(i) = \sum_{l \in (T_{k+1}, \overline{T}_{k+1})} C_l^{-1}(z_{k+1}), \tag{9}$$

then $z_k < z_{k+1} \leq z^*$.

It is important to note that the cuts $[T_{k+1}, \overline{T}_{k+1}]$ thus
generated are all different from each other since if we have
$T_l = T_k$ for some $k \neq l$, then by (9) we must have

$$\sum_{l \in (T_k, \overline{T}_k)} C_l^{-1}(z_k) = \sum_{l \in (T_k, \overline{T}_k)} C_l^{-1}(z_l),$$

which is impossible since both z_k and the cost functions
are strictly increasing. Since the number of cuts in G is
finite, the above procedure must stop in a finite number of
steps. By the end of the procedure the optimal solutions is
found.

From the above discussion we have the following generic
algorithm for solving problem B.
algorithm C:
INPUT: A network $G = (N, A)$, link cost functions $C_l(x)$ for
all $l \in A$, node commodities $b(i)$, for all $i \in N$.
OUTPUT: A link load vector x^* solving Problem B.
```
min_max(G, C, b):
  solve ∑_{l∈(N_s,N̄_s)} C_l^{-1}(z_0) = ∑_{i∈N_s} b(i);
  for k = 0 to ∞ do
    form network Ĝ_k with link capacities u_l = C_l^{-1}(z_k), l ∈
A;
    x_k =maximum_flow(Ĝ_k).argument;
    if (all links (s, i), i ∈ N_s are saturated) then
        do value = z_k;
           argument= x_k;
          return(value,argument);
    else do
        find a minimum cut[{s} ∪ T_{k+1}, {s} ∪ T_{k+1}] of
Ĝ_k;
        solve
```

$$\sum_{l \in (T_{k+1}, \overline{T}_{k+1})} C_l^{-1}(z_{k+1}) = \sum_{i \in T_{k+1}} b(i);$$

```
  end;
return;
```
While the algorithm thus obtained is guaranteed to ter-
minate in a finite number of steps, it may take a long time

since the number of candidate cuts may exponential in the size of the network. However, as we will see next, in the important case of piecewise linear costs, it can be shown that the algorithm terminates in polynomial time. It will facilitate the discussion if we consider first linear cost functions.

Linear Cost Functions.

Define $F_{T_k} = \sum_{i \in N_s - T_k} b(i) - \sum_{i \in N_t \cap T_k} b(i)$. Then we have the following lemma

Lemma 8: If the link costs are linear, $C_l(x) = x/c_l$, and $z_k < z^*$ then

$$F_{T_k} < F_{T_{k+1}}, \ k \geq 1.$$

Assume now that for the maximum flow problem in Algorithm C, an algorithm based on augmenting paths on the residual network (see [2] for the definition) is used, and as an initial flow at iteration $k+1$, the flow $x^{(k)}$ is used. Since the links

$$L_k = \{l : l = (s, i), i \in (N_s - T_k),$$
$$\text{or } l = (i, t), i \in (N_t \cap T_k)\},$$

in $\widehat{G}_k$ are all loaded to capacity by $x^{(k)}$, it can be seen that these links will all be loaded to capacity in $\widehat{G}_{k+1}$ by flow $x^{(k+1)}$. Taking into account Lemma 8 we conclude that L_{k+1} is a strict superset of L_k. This means that at each iteration for $k \geq 1$, at least one of the links in the set

$$L = \{l : l = (s, i), \ i \in N_s, \text{ or } l = (i, t), \ i \in N_t\}$$

is loaded to capacity. Hence the algorithm will complete at most in $|N_s| + |N_t|$ iterations. We conclude that for linear link costs, the complexity of Algorithm C is $O((|N_s| + |N_t|) |N|^3)$, or in terms of the number of nodes, $O(|N|^4)$. In fact, if $|N_s| = |N_t| = 1$, then Algorithm C can terminate in one iteration. Indeed, in this case either $x^{(0)}$ is the optimal flow, or the cut $[T_1, \overline{T}_1]$ is separating the nodes $N_s = \{i_s\}$ and $N_t = \{i_t\}$, and since it is a minimum cut for $\widehat{G}_0$, we have,

$$x_l^{(0)} = c_l z_0, \text{ for all } l \in (T_1, \overline{T}_1) \text{ and}$$
$$x_l^{(0)} = 0 \text{ for all } l \in (\overline{T}_1, T_1) \tag{10}$$

Moreover,

$$x_l^{(0)} \leq c_l z_0, \text{ for all } l \in A. \tag{11}$$

Set now

$$z_1 = \frac{b(i_s)}{\sum_{l \in (T_1, \overline{T}_1)} c_l} \tag{12}$$

and

$$x^{(1)} = \frac{z_1}{z_0} x^{(0)},$$

From (10), (11), (12) and Lemma 3 b) it follows that $x^{(1)}$ solves problem B.

The difference between linear and nonlinear costs can be seen in the example in Figure 4. For linear costs, the algorithm terminates in one iteration. If, however, the link
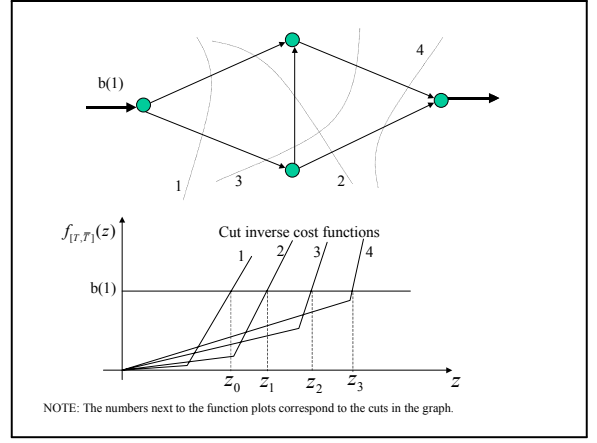

Fig. 4. Algorithm C for non-linear costs

cost functions are such that the resulting cut "inverse" cost functions, i.e., the functions

$$f_{[T, \overline{T}]}(z) = \sum_{l \in (T, \overline{T})} C_l^{-1}(z)$$

are the ones plotted in the figure, then the algorithm examines all 4 cuts before termination.

Complexity of Algorithm A for linear costs. The complexity of algorithm C for linear costs is $O(|N|^4)$. If $f(|N|)$ is the number of computations needed for Algorithm A and $\phi(|N|)$ is the worst-case number of computation needed for Algorithm C, then we have for some k_1, $1 \leq k_1 \leq f(\lfloor |N|/2 \rfloor)$

$$f(|N|) \leq \phi(|N|) + f(|k_1|) + f(|N| - |k_1|) \tag{13}$$

It is easy to see by induction from (13) that $f(|N|) \leq |N| f(1) + \sum_{k=1}^{|N|} \phi(|k|)$ and since $\phi(|N|) = O(|N|^4)$, we have that $f(|N|) = O(|N|^5)$. The complexity of Algorithm B is the same as algorithm A.

Piecewise Linear Cost Functions

Assume that the links costs are of the form

$$C_l(x) = \max_{j=1,..,M_l} \left\{ \frac{x}{c_{l,j}} + \sigma_{l,j} \right\}, \ c_{l,1} > c_{l,2}, ... > c_{l,M_l}, \tag{14}$$

$$\sigma_{l,1} = 0 > \sigma_{l,2} > ... > \sigma_{l,M_l}.$$

For $z > 0$, let $x/\widehat{c}_l(z) + \widehat{\sigma}_l(z)$ be the "linear segment of $C_l(x)$ at z", i.e.,

$$z = x/\widehat{c}_l(z) + \widehat{\sigma}_l(z),$$
$$(\widehat{c}_l(z), \widehat{\sigma}_l(z)) \in \{(c_{l,1}, \sigma_{l,1}), ..., (c_{l,M_l}, \sigma_{l,M_l})\}.$$

With a slight modification of the proof of Lemma 8 it can be proved that

Lemma 9: If the link costs are piecewise linear as in (14) and $z_k < z^*$, then either $\widehat{c}_l(z_{k+1}) < \widehat{c}_l(z_k)$ for at least one l, or

$$F_{T_k} < F_{T_{k+1}}, \ k \geq 1.$$

Arguing as in the case of linear costs, we conclude that if $F_{T_k} < F_{T_{k+1}}$ then at least one of the links in the set

$$L = \{l : l = (s,i),\ i \in N_s,\ \text{or}\ l = (i,t),\ i \in N_t\}$$

will be loaded to capacity. According to Lemma 9, at iteration $k+1$, either at least one of the links in the set L is loaded to capacity, or at least one of the slopes of the linear segments of $C_l(x)$, at z_k, $l \in A$, is increasing. Therefore, the algorithm will terminate in the worst case in $\sum_{l \in A}(M_l - 1) + |N_s| + |N_t|$ iterations. We conclude that the complexity of Algorithm C in this case is $O((\sum_{l \in A} M_l + |N_s| + |N_t|)|N|^3)$, or in terms of the number of nodes and links, $O(|N|^4 + |N|^3 |A|\, M)$, where $M = \max_{l \in A}\{M_l\}$. Arguing as for the case of linear cost functions, we see that the complexity of Algorithms A and B is $O(|N|^5 + |N|^4 |A|\,(M-1))$.

IV. NUMERICAL RESULTS

In this section we present numerical results regarding the computational efficiency of the algorithm presented in Section III. We created random networks with fixed number, N, of nodes as follows.

The nodes are numbered from 1 to N. We select uniformly K random variables, $K \in \{1, ..., N\}$ and then, for $k = 1, ..., K$, we select a random sequence of n_k distinct numbers $i_1^k, i_2^k, ..., i_{n_k}^k$, where $i_m^k \in \{1, ..., N\}$. Node i_1^k is considered a source node and node $i_{n_k}^k$ a destination node. We connect node i_m^k to i_{m+1}^k, if they are not connected already. Hence, with the above process we create a path from source i_1^k to destination $i_{n_k}^k$. Note that using this process it is possible that a node is assigned both as a source and as a destination. We resolve this issue below.

Next we assign numbers to flows for the source nodes by picking random numbers between 1 and 120. Once these flows are assigned, we pick flows for the destination nodes. These latter flows are assigned so that there is always a feasible solution to the problem at hand. This is done as follows. For each source node, we find the corresponding reachable destination nodes. Note that by the construction of the network for every source node there is always a reachable destination. We split the flow of the source node to the destinations randomly. For each destination node we add all flows that have been assigned in this manner. Finally, if a node has been assigned both as a source and a destination, we subtract the output flow from the input flow of that node. If the result is positive (negative), the node is a source (destination) node. If the result is zero, the node is a transit node. Hence, according to the above process the number of source (destination) nodes is at most K.

Finally we assign capacities to each of the links of the network by picking randomly numbers between 1 and 120. As cost functions we consider the link utilizations, $C_l(x_l) = x_l/B_l$, where B_l is the physical capacity of link l.

We run the simulations on a 300MHz PC. For each network size, we created 10 networks. For algorithms A and

Nodes	Links min - max	Alg. A	Alg. B
60	146-355	2,4	2
70	173-478	4,45	3,64
80	181-563	5,44	4,93
90	199-691	7,4	6,89
100	223-814	8,3	7,67
110	246-920	12,9	12,7
120	317-1070	21,24	19,7
130	271-1172	23,65	20,4
140	243-1214	28,1	24,6
150	403-1452	31,2	26,8

TABLE I

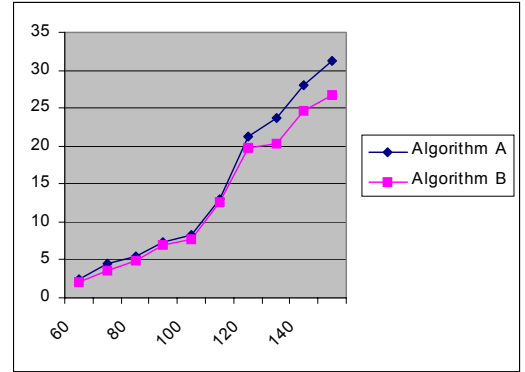RUNNING TIMES OF ALGORITHMS A AND B

Fig. 5. Performance Comparison of Algorithms A and B

B, Table I shows, for each network size, the minimum and maximum number of links of the networks selected and the average run time in seconds. Recall that at each iteration algorithm A splits the network in two, while algorithm B splits it in a number that depends on the strongly connected components of the modified network. We observe that for either algorithm the performance is very satisfactory even for fairly large networks. Regarding the comparison between the two algorithms, we see (see also Figure 5) that as expected, Algorithm B performs better. The difference in performance is more significant for larger number of nodes. A probable reason for this observation is that the number of strongly connected components in the random networks created is rarely more than two when the number of nodes is small.

V. CONCLUSIONS

We provided an algorithm to determine the lexicographically optimal bandwidth allocation in order to satisfy the communication needs between two private networks. The algorithm is guaranteed to converge in finite number of steps. For piecewise linear costs its complexity is $O(|N|^5 + |N|^4 |A|\,(M-1))$, where M is the maximum number of linear segments that a cost function may contain.

Numerical results show that the algorithm is computationally efficient even for fairly large networks. While there are various steps in the proposed algorithm, each of the steps is based on algorithms (max-flow, min-cut) that are easily available. Hence its implementation is simple.

As was mentioned in the introduction, the problem we considered addresses only one of the many issues related to bandwidth allocation and can serve as a building block for them. We indicate in the following some topics that need further investigation.

• The generalization of the problem we addressed is to consider the communication needs of multiple private networks and to provide a solution for the associated multicommodity lexicographically optimal bandwidth allocation problem. Note that the algorithm as it stands now, can also solve the following problems. a) given an ingress node and multiple egress nodes with bandwidth demands for each egress node, to find the lexicographically optimal flow satisfying the demands, b) given an egress node and multiple ingress nodes with given bandwidth demands from each of the ingress nodes to the egress node, to find the lexicographically optimal flow satisfying the demands. However, the generalization to multiple ingress and egress nodes with specific demands for each ingress-egress pair needs further work.

• The solution provided is noninteger, which is a good assumption when the application needs are much smaller than the total requested bandwidth (e.g., voice 64kbps versus total in the range of Mbps). In case the application needs are comparable to the requested bandwidth and all the application traffic needs to be routed through a fixed path (connection oriented network), then integer solutions should be sought.

• In integrated networks where applications with varying Quality of Service (QoS) requirements may exist in the network, it is desirable to impose additional constraints on the paths followed through the backbone network. In this case, the optimization problem should take these QoS constraints into account.

• We addressed in this paper the dimensioning problem of providing in a centralized fashion optimal routes, given requested or anticipated demands. These routes can be used during the real-time operation of the network in order to route connection requests. The real-time operation of the network must further take into account demand fluctuations above or below the anticipated demands. In this respect, it is important to investigate the possibility of implementing in a distributed fashion the proposed algorithms and to study their convergence.

REFERENCES

[1] R. K. Ahuja, "Algorithms for the Minimax Transportation Problem," *Naval Research Logistics Quarterly,* **33,** pp 725-740, 1986.

[2] R. K. Ahuja, T. L. Magnati, J. B. Orlin, *Network Flows, Theory, Algorithms, and Applications,* Prentice Hall, 1993.

[3] G. R. Ash, R. H. Cardwell and R. P. Murray, "Design and Optimization of Networks with Dynamic Routing," *Bell Syst. Tech. J.*, vol 60, pp. 1787-1820, 1981.

[4] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell and J. McManus, "Requirements for Traffic Engineering Over MPLS," *Internet Draft, RFC2702,* September 1999

[5] D. Bertsekas, *Nonlinear Programming,* Athena Scientific, 1995.

[6] D. Bertsekas, R. Gallager, *Data Networks,* Prentice Hall, 1992.

[7] C. S. Chang, "Stability Queue Length and Delay of Deterministic and Stochastic Queueing Networks," *IEEE Transactions on Automatic Control*, Vol 39, 913-931, 1994.

[8] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method: An approach to store-and-forward communication network design," *Networks,* vol 3, pp 331-340, 1978.

[9] A. Girard, *Routing and Dimensioning in Circuit-Switched Networks.* Readin, MA: Addison-Wesley, 1988.

[10] A. Girard and B. Sanso, "Multicommodity Flow Models, Failure Propagation, and Reliable Loss Network Design," *IEEE/ACM Transactions on Networking*, Vol 6, No 1, February 1998.

[11] R. J. Gibbens and P. J. Hunt, "Effective Bandwidths for multitype UAS Channel," *Queueing Systems*, Vol 9, 17-28, 1991.

[12] A. Greenberg, R. Srikant, "Computational Techniques for Accurate Performance Evaluation of Multirate Multihop Communication Networks," *IEEE/ACM Trans. on Networking*, vol 5, no 2, April 1997, pp 266-277.

[13] R. Guerin, H. Ahmadi, N. Naghshineh, "Equivalent Capacity and its Applications to Bandwidth Allocation in High-speed Networks," *IEEE J. Select. Areas in Commun.*, Vol. 9, 968-981, 1991.

[14] F. P. Kelly, "Loss Networks," *Ann. App. Prob.* vol 1, pp 319-378, Aug 1991.

[15] F. P. Kelly, "Effective Bandwidths at multi-class queues," *Queueing Systems,* Vol. 9, 5-16, 1991.

[16] A. Kerhenbaum, *Telecommunications Network Design Algorithm,* New York: Mac Graw-Hill, 1993.

[17] L. Kleinrock, *CommunicationNets:Stochastic Message Flow and Delay.* New York: McGraw-Hill, 1964.

[18] L. Li and A. K. Somani, "Dynamic Wavelength Routing Using Congestion and Neighborhood Information," *IEEE/ACM Transactions on Networking,* Vol. 7, No 5, 779-786, 1999.

[19] D. Medhi and S. Guptan, "Network Dimensioning and Performance of Multiservice, Multirate Loss Networks with Dynamic Routing," *IEEE/ACM Transactions on Networking,* vol 5, no 6, December 1997, pp 944-957.

[20] Y. T'Joens, P. Georgatsos, D. Griffin, P-T. Huth. L. Georgiadis, G. Pavlou, D. Manikis, E. Mykoniati, "An Integrated Approach to Switched VC ATM Restoration in the REFORM System", *Int. Conf. on Design of Reliable Communication Networks, DRCN'2000,* 10-12 April 2000, Munich, Germany.

[21] Y. Wang, Z. Wang, L. Zhang, "Internet Traffic Engineering Approaches and Algorithms," *Networld+Interop,* Las Vegas, May 2000.

EXISTENCE OF LEXICOGRAPHICALLY OPTIMAL POINT

The existence of lexicographically optimal point can be established under very general conditions. We provide here an outline of the argument.

Let $(y_1, ... y_K)$ be a K-dimensional vector and let k_i, $i = 1, 2, ...K$ be such that

$$y_{k_1} \geq y_{k_2} \geq ... \geq y_{k_K}.$$

Define

$$\overset{(i)}{\max} \{y_1, ..., y_k\} = y_{k_i}.$$

It can be seen that $\max^{(i)} \{y_1, ..., y_K\}$ is a continuous function of the vector $(y_1, ..., y_K)$.

Consider next a set $A \neq \varnothing$ in an N-dimensional space, and let $f_k(x)$, $k = 1, 2, ..., K$ be real functions defined on A. Let $A_0 = A$ and define A_i, $i = 1, 2, ...K$ recursively as follows: A_i is the set of solutions to problem

$$\min_{x \in A_{i-1}} \overset{(i)}{\max} \{f_1(x), ..., f_k(x)\}.$$

By definition it holds $A_i \subseteq A_{i-1}$. Also, using the definition of the lexicographic order, it can be seen that a point x^* in A induces the lexicographically optimal vector $(f_1(x), ..., f_k(x))$ among all vectors in A, if and only if x^* solves

$$\min_{x \in A_{i-1}} \overset{(i)}{\max} \{f_1(x), ..., f_k(x)\},$$

for all $i = 1, 2, ...K$. In other words, the set of points that induce the lexicographically optimal vector $(f_1(x), ..., f_k(x))$ among all vectors in A is $A^* = A_K$.

Assume now that A is compact and $f_k(x)$ are continuous for all $k = 1, 2, ...K$. Since $\max^{(i)} \{y_1, ..., y_K\}$ is a continuous function of vector $(y_1, ...y_K)$, it follows that

$$\overset{(i)}{\max} \{f_1(x), ..., f_k(x)\}$$

is a continuous function of x. Since A is compact and nonempty a solution to problem

$$\min_{x \in A} \overset{(i)}{\max} \{f_1(x), ..., f_k(x)\}$$

exists and A_1 is compact and nonempty [5, Appendix A.2]. By induction we conclude that each A_i, $i = 2, ..., K$, is compact and nonempty and hence A^* is also compact and nonempty.

PROOF OF LEMMA 2

Since $C_l(x_l^*) \leq \beta$, and the cost functions are increasing, we have $x_l^* \leq C_l^{-1}(\beta) = u_l(\beta)$ for all $l \in A$. We then have

$$b(S) = \sum_{(i,j) \in (S, \overline{S})} x_{ij}^* - \sum_{(i,j) \in (\overline{S}, S)} x_{ij}^*$$

$$\leq \sum_{(i,j) \in (S, \overline{S})} u_{ij}(\beta) = U_\beta[S, \overline{S}]$$

for every set $S \subseteq N$. Assume that for every subset $S \subseteq N$ with $b(S) > 0$ we have $b(S) < U_\beta[S, \overline{S}]$. Then, since the cost functions $C_l(x)$ are continuous and strictly increasing, and $\beta > 0$ (recall the assumption $N_s \neq \emptyset$), we can pick $\varepsilon > 0$ small enough so that $b(S) \leq U_{\beta - \varepsilon}[S, \overline{S}]$ for every $S \subseteq N$ with $b(S) > 0$. Since for all sets with $b(S) = 0$ we obviously have $0 = b(S) \leq U_{\beta - \varepsilon}[S, \overline{S}]$, Theorem 1 implies that there is a feasible flow for the network with capacities $u_l(\beta - \varepsilon)$, $l \in N$, which in turn implies that there is a flow whose maximal link cost is at most $\beta - \varepsilon$. However, this contradicts the fact that β is the optimal value of problem B. Therefore, there must be a set T with $b(T) > 0$ such that $b(T) \geq U_\beta[T, \overline{T}]$, which together with the fact that $b(S) \leq U_\beta[S, \overline{S}]$ for all $S \subseteq N$ implies that $0 < b(T) = U_\beta[T, \overline{T}]$. ∎

PROOF OF LEMMA 3

Since x' solves Problem B, as in the proof of Lemma 2, we have,

$$b(T) = \sum_{(i,j) \in (T, \overline{T})} x_{ij}' - \sum_{(i,j) \in (\overline{T}, T)} x_{ij}'$$

$$\leq \sum_{(i,j) \in (T, \overline{T})} u_{ij}(\beta) = U_\beta[T, \overline{T}].$$

If any of the values $x_{ij}' \in (\overline{T}, T)$ are positive, the last inequality would be strict, a contradiction. Therefore, we have $x_{ij}' = 0$ for every $(i, j) \in (\overline{T}, T)$ and

$$b(T) = \sum_{(i,j) \in (T, \overline{T})} x_{ij}' \leq U_\beta[T, \overline{T}].$$

If $C_{l_0}(x_{l_0}') < \beta$ for some $l_0 \in (T, \overline{T})$, then taking into account that $C_l(x)$ is strictly increasing, we would have $x_{l_0}' < C_{l_0}^{-1}(\beta) = u_{l_0}(\beta)$. Since $x_l' \leq u_l(\beta)$ for all $l \in A$, and the set $(T, \overline{T})$ is nonempty (by virtue of the fact that $b(T) > 0$) we would have

$$b(T) = \sum_{(i,j) \in (T, \overline{T})} x_{ij}' < U_\beta[T, \overline{T}],$$

a contradiction.

To show the converse, notice first that from (3) and (4), it follows immediately that $b(T) = U_\beta[T, \overline{T}]$. Assume that x' is not optimal flow for problem B. Let $0 < \beta^{(1)} < \beta$ be the optimal value of Problem B and let $x^{(1)}$ be an optimal flow. Taking into account that the set $(T, \overline{T})$ is nonempty and the cost functions are strictly increasing, we then have

$$b(T) = \sum_{(i,j) \in (T, \overline{T})} x_{ij}^{(1)} - \sum_{(i,j) \in (\overline{T}, T)} x_{ij}^{(1)}$$

$$\leq \sum_{(i,j) \in (T, \overline{T})} C_{ij}^{-1}(\beta^{(1)})$$

$$< \sum_{(i,j) \in (T, \overline{T})} C_{ij}^{-1}(\beta) = U_\beta[T, \overline{T}],$$

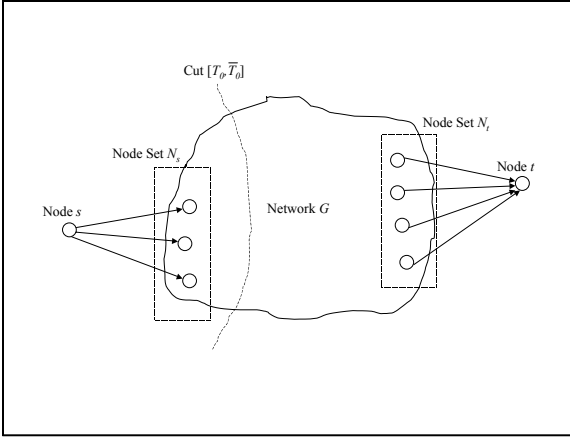a contradiction (the last inequality is strict since the set $(T, \overline{T})$ is nonempty). ∎
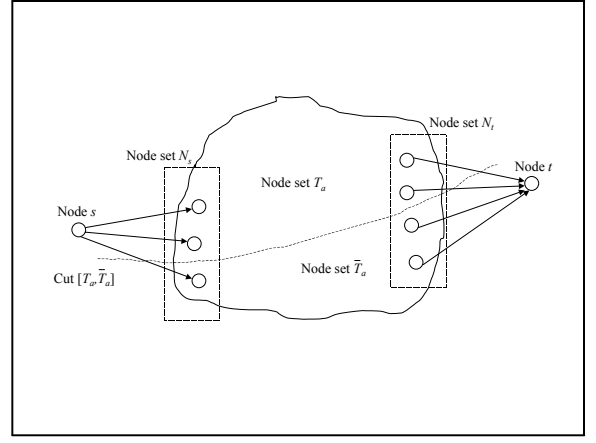
Fig. 6. Augmented Network

Fig. 7. The Minimum Cut of the Augmented Network

PROOF OF LEMMA 5

Let l be a link with, say, tail in $\widetilde{G}_1$ and head in $\widetilde{G}_2$. Then there is a partition $\Pi_1 = \left\{ \widetilde{G}_i, \ i \in s_1 \right\}$, $\Pi_2 = \left\{ \widetilde{G}_i, \ i \in s_2 \right\}$ of $\widetilde{G}_1, \widetilde{G}_2, \ldots, \widetilde{G}_L$, with $\widetilde{G}_1 \in \Pi_1$ and $\widetilde{G}_2 \in \Pi_2$ such that all links connecting $S_1 = \cup_{i \in s_1} \widetilde{G}_i$ to $S_2 = \cup_{i \in s_2} \widetilde{G}_i$ have tail in S_1 and head in S_2. To see this, consider first the partition $\Pi_1 = \left\{ \widetilde{G}_1 \right\}$, $\Pi_2 = \left\{ \widetilde{G}_i, \ i \in \{2, \ldots L\} \right\}$, and define $S_1 = \widetilde{G}_1$, $S_2 = \cup_{i \in \{2, \ldots, L\}} \widetilde{G}_i$. If all links in $[S_1, S_2]$ are not outgoing, select the components in Π_2 which are connected with S_1 with links having head in $\widetilde{G}_1$, and move them to partition Π_1. If again all links in $[S_1, S_2]$ are not outgoing, select the components in Π_2 which are connected with S_1 with links having head in one of the sets in Π_1, and move them to Π_1. Note that by construction *there is a path connecting all components in Π_1 to component $\widetilde{G}_1$*. Proceeding in this fashion, we either find a partition for which all links in $[S_1, S_2]$ are outgoing, or, since the number of components is finite, we will have to include $\widetilde{G}_2$ to Π_1 at some step. The latter case, however is impossible since including the link l to the path connecting $\widetilde{G}_2$ to $\widetilde{G}_1$, we create a circle, which implies that there is a path from every node of $\widetilde{G}_1$ to every node in $\widetilde{G}_2$ and vice versa, a contradiction.

Let now $[S_1, S_2]$ be the cut determined with the previous procedure, having only forward links. Note that each of the links in the cut $[S_1, S_2]$ must be either maximally loaded or zero loaded, since by definition all other links in $\widetilde{G}$ are bidirectional. Moreover, all maximally loaded links are forward links and all zero loaded links are backward link of the cut $[S_1, S_2]$ in network G. If there are no maximally loaded links in $[S_1, S_2]$, then there are only backward links of $[S_1, S_2]$ in network G, and moreover, since these links are zero loaded, we have that $b(i) = 0$ for all $i \in S_1$. Hence any flow on G will have zero load on all the links in S_1 and in $[S_1, S_2]$. If some of the links in $[S_1, S_2]$ are maximally loaded, then the cut $[S_1, S_2]$ is a maximally loaded cut ∎

PROOF OF LEMMA 6

Augment network G to $\widehat{G}_0$, with two nodes s, t, and with arcs (s, i), $i \in N_s$, (i, t), $i \in N_t$ (see Figure 6). The capacity of an arc (s, i), $i \in N_s$ is $b(i)$ and the capacity of an arc (i, t), $i \in N_t$ is $-b(i)$. Assign capacities $u_l = C_l^{-1}(z_0)$, $l \in A$ and solve the maximum flow problem on $\widehat{G}_0$ to obtain a value V_0 and an optimal flow $x^{(0)}$. Note that by the definition of the augmented network $\widehat{G}_0$ we have $V_0 \leq \sum_{i \in N_s} b(i)$. Therefore, we consider two cases.

a) $V_0 = \sum_{i \in N_s} b(i)$. Then by (8) we also have $V_0 = U_{z_0}[T_0, \overline{T_0}]$. That is, the cut $[T_0, \overline{T_0}]$ is minimum under the assigned capacities, which implies that $x_l^{(0)} = C_l^{-1}(z_0)$, $l \in (T_0, \overline{T}_0)$ and $x_l^{(0)} = 0$, $l \in (\overline{T}_0, T_0)$. Hence this cut satisfies the conditions of Lemma 3 b) and therefore $z_0 = z^*$.

b) $V_0 < \sum_{i \in N_s} b(i)$. Then there must be a minimum cut $[T, \overline{T}]$ in $\widehat{G}_0$ such that $T \cap N \neq \emptyset$. Let this minimum cut be $\left[\{s\} \cup T_a, \overline{\{s\} \cup T_a} \right]$, where $T_a \subset N$ (see Figure 7). Then $[T_a, \overline{T}_a]$ is a cut in G. Moreover, $(T_a, \overline{T}_a) \neq \emptyset$. To see this note that if $(T_a, \overline{T}_a) = \emptyset$ then since all links in $\left(\{s\} \cup T_a, \overline{\{s\} \cup T_a} \right)$ are maximally loaded we would have

$$V_0 = \sum_{i \in N_s - T_a} b(i) - \sum_{i \in T_a \cap N_t} b(i) < \sum_{i \in N_s} b(i),$$

hence

$$-\sum_{i \in T_a \cap N_t} b(i) < \sum_{i \in N_s \cap T_a} b(i).$$

However, this is impossible since the facts that problem B has a solution and there are no links connecting T_a and $\overline{T}_a$ (by the assumption $(T_a, \overline{T}_a) = \emptyset$), imply by the conservation of flow that

$$-\sum_{i \in T_a \cap N_t} b(i) = \sum_{i \in N_s \cap T_a} b(i).$$

Consider now Problem B on network G, but with commodities $b^{(0)}(i)$ defined as follows.

$$b^{(0)}(i) = x_{si}^{(0)} = \begin{cases} b(i) & \text{if } i \in N_s - T_a \\ x_{si}^{(0)} \leq b(i) & \text{if } i \in N_s \cap T_a \end{cases}$$

$$b^{(0)}(i) = -x_{it}^{(0)} = \begin{cases} b(i) & \text{if } i \in T_a \cap N_t \\ -x_{it}^{(0)} \geq b(i) & \text{if } i \in N_t - T_a \end{cases}$$

$$b^{(0)}(i) = 0 \text{ elsewhere}$$

Since $x^{(0)}$ is the maximum flow, we have $x_l^{(0)} = C_l^{-1}(z_0)$, $l \in (T_a, \overline{T}_a)$ and $x_l^{(0)} = 0, l \in (\overline{T}_a, T_a)$. Since $(T_a, \overline{T}_a) \neq \emptyset$, it follows from Lemma 3 b) that $x^{(0)}$ solves problem B for network G with commodities $b^{(0)}(i)$ and the optimal value is z_0. Since $b^{(0)}(i) \leq b(i)$, $i \in N_s$, and $b^{(0)}(i) \geq b(i)$, $i \in N_t$, it follows that $z_0 \leq z^*$. $\blacksquare$

PROOF OF LEMMA 7

If $x^{(k)}$, saturates all links (s, i), $i \in N_s$, then $x^{(k)}$ is a feasible flow of problem B and since $x_l^{(k)} \leq C_l^{(-1)}(z_k)$, $l \in A$, and z^* is the optimal value of problem B, we have,

$$z^* \leq \max_{l \in A} \left\{ C_l \left(x_l^{(k)} \right) \right\} \leq z_k.$$

Since by assumption $z_k \leq z^*$ we conclude that $z_k = z^*$, and we have found an optimal solution.

If $x^{(k)}$ does not saturate all links (s, i), $i \in N_s$, then as with the initial value z_0, there must be a cut $[T_{k+1}, \overline{T}_{k+1}]$ in G such that $(T_{k+1}, \overline{T}_{k+1}) \neq \emptyset$ and $\left[\{s\} \cup T_{k+1}, \overline{\{s\} \cup T_{k+1}} \right]$ is a minimum cut in $\widehat{G}_k$. Also, since not all links (s, i), $i \in N_s$ are saturated, the maximum flow in $\widehat{G}_k$ is smaller than $\sum_{i \in N_s} b(i)$. Hence,

$$\sum_{i \in N_s} b(i) > \sum_{i \in N_S - T_{k+1}} b(i) + \sum_{l \in \left(T_{k+1}, \overline{T}_{k+1} \right)} C_l^{-1}(z_k)$$
$$- \sum_{i \in N_t \cap T_{k+1}} b(i)$$
$$\geq \sum_{l \in \left(T_{k+1}, \overline{T}_{k+1} \right)} C_l^{-1}(z_k).$$

Hence,

$$\sum_{i \in T_{k+1}} b(i) > \sum_{l \in \left(T_{k+1}, \overline{T}_{k+1} \right)} C_l^{-1}(z_k). \qquad (15)$$

Since z_{k+1} is the solution to the equation

$$\sum_{i \in T_{k+1}} b(i) = \sum_{l \in \left(T_{k+1}, \overline{T}_{k+1} \right)} C_l^{-1}(z_{k+1}). \qquad (16)$$

and the cost functions are increasing, it is clear from (15) and (16) that $z_{k+1} > z_k$. The fact that $z_{k+1} \leq z^*$ follows as in the case of z_0 in Lemma 6.

PROOF OF LEMMA 8

Since $\left[\{s\} \cup T_k, \overline{\{s\} \cup T_k} \right]$ is a minimum cut of $\widehat{G}_{k-1}$ we have

$$F_{T_k} + \left(\sum_{l \in (T_k, \overline{T}_k)} c_l \right) z_{k-1} \leq F_{T_{k+1}} + \left(\sum_{l \in (T_{k+1}, \overline{T}_{k+1})} c_l \right) z_{k-1}$$
$$(17)$$

On the other hand, if $z_k < z^*$ then x_k does not saturate all links (s, i), $i \in N_s$. Since $\left[\{s\} \cup T_{k+1}, \overline{\{s\} \cup T_{k+1}} \right]$ is a minimum cut in $\widehat{G}_k$ and by definition

$$\left(\sum_{l \in (T_k, \overline{T}_k)} c_l \right) z_k = \sum_{i \in T_k} b(i),$$

we have

$$F_{T_k} + \left(\sum_{l \in (T_k, \overline{T}_k)} c_l \right) z_k = \sum_{i \in N_s} b(i) > F_{T_{k+1}}$$
$$+ \left(\sum_{l \in (T_{k+1}, \overline{T}_{k+1})} c_l \right) z_k \qquad (18)$$

Subtracting (18) from (17) we have

$$\left(\sum_{l \in (T_k, \overline{T}_k)} c_l \right) (z_{k-1} - z_k) < \left(\sum_{l \in (T_{k+1}, \overline{T}_{k+1})} c_l \right) (z_{k-1} - z_k)$$

and since $z_k > z_{k-1}$,

$$\sum_{l \in (T_k, \overline{T}_k)} c_l > \sum_{l \in (T_{k+1}, \overline{T}_{k+1})} c_l.$$

It follows from (17) that $F_{T_k} < F_{T_{k+1}}$. $\blacksquare$