Virtual Topologies for Topology Abstraction Service for IP-VPNs

Lavanya Sivakumar
SRM Research Institute
SRM University
Kattankulathur, Chennai - 603203, INDIA.
Email: $lavanya.s@res.srmuniv.ac.in$

Jayaram Balabaskaran
School of Computing Sciences
VIT University
Chennai - 600127, India.
Email: $jayaram.b@vit.ac.in$

Krishnaiyan Thulasiraman
School of Computer Science
University of Oklahoma
Norman, OK 73019, U.S.A.
Email: $thulasi@ou.edu$

S. Arumugam
National Center for Advanced Research in Discrete Mathematics
Kalasalingam University
Anand Nagar, Krishnankoil-626126, INDIA.
Email: $s.arumugam.klu@gmail.com$

***Abstract*—Research in topology abstraction(TA) service for IP-VPNs was initiated in [2] and has been studied in a series of papers [3], [11] and [5]. In the more recent work[5], three decentralized schemes to generate topology abstractions were proposed and evaluated through extensive simulations. These schemes used some very powerful tools from graph theory and combinatorial optimization. But the topology considered in [5] for abstraction is very simple and sparse. In this paper we continue this study and propose two new schemes for topology abstraction. Specifically, given a core network with node set V, and also given the node-to-node maximum flow matrix $F = [f(u, v)]$ for any connected graph $G = (V^*, E)$ with V^* a subset of V, we show how to assign capacities to the edges of G such that the maximum available flow between any pair of vertices $u, v \in V^*$ in G is at most the maximum available flow between u and v in the core network. Though this methodology is applicable to any graph, we seek virtual topologies with certain other desirable properties: low degree, number of edges being linear in the number of vertices, amenable to survivable logical topology routing in an IP over WDM optical networks etc. Chordal graphs have such properties. In this paper we show how to constructed $2-$vertex and $2-$edge connected chordal graphs, starting from the Gomory-Hu tree of the flow matrix of the core network.**

I. Introduction

VPN service providers(VSP) and IP-VPN customers have maintained service demarcation boundaries between their routing and signaling entities resulting in the VPN's viewing the VSP network as an opaque entity and therefore limiting interaction between the two. Ravi et. al in [2] introduced the notion of topology abstraction as a means for sharing the core topology information as abstract graphs associated with QoS metric information. The authors in [2] study the abstraction of the available capacity associated with the links of the VSP's core network. The VSP generates the TA for each VPN based on the TA SLA parameters, so that the VSP can share its core topology in a manner that is practical and scalable through the means of an abstraction that captures the properties of the core topology. This concept was further explored in a series of papers [3], [11], [5]. Other related works may be found in [4], [12], [13], [14], [15] and [16].

In [5] the authors proposed three decentralized schemes to generate topology abstractions. In these schemes, topology abstraction for each VPN is generated using the same core network information. Specifically, these schemes assume that the border nodes(connecting the VPNs to the core network) performing abstractions have access to the entire core topology and generate TA's to its hosted VPNs independent of one another. It has been observed that this form of decentralized TA generation leads to insufficient resource utilization and poor call performance. To overcome this problem in a recent work a decentralized scheme has been proposed([6]).

The topology used in [5] and [6] is very simple and sparse. In this paper we use chordal graphs as a virtual topology and propose an abstraction scheme that is based on the following properties:

1. Connectivity of a graph.
2. Sparseness of a graph.
3. Maximum flow in a graph.

In the proposed TA scheme we ensure that the resulting TA is at least $2-$connected, has fewer edges and

also maintains the maximum flow property between all pairs of vertices from the original graph. Additionally, the resulting TA topology is not a sub-graph of the original graph. More formally, we propose an algorithm to construct a $2-$edge and $2-$vertex connected chordal graph G as a logical topology for the IP-VPN abstraction problem. We also show that, for any pair of vertices, the maximum flow in the new graph will be less than or equal to the maximum flow in the core network. We use the concepts of Gomory-Hu trees[18] and the fundamental cut-set based approach([17]).

The rest of the paper is organized as follows. In section 2 we review Gomory-Hu trees, fundamental cutsets and give necessary notations used throughout the paper. Given the maximum flows between all pairs of vertices of the core network and the corresponding Gomory-Hu tree ,(to be described later) capturing this flow information, and also a graph G with T as a spanning tree of G, we discuss an algorithm in section 3 to assign capacities to the edges of G so that the maximum flow between any pair of vertices u and v in G is at most the maximum flow between u and v in the core network. The assumption that G has T as a spanning tree does not restrict the validity of the capacity assignment. For, if the given T tree is not present in G, then one can add pseudo edges to G representing the edges of T. In section 4 we first give a basic algorithm to construct a 2-edge connected graph from any tree T. We then make a modification of the basic algorithm to construct a $2-$edge and $2-$vertex connected chordal graph from the Gomory-Hu tree. Combining the results of sections 3 and 4 one can construct a $2-$ edge connected $2-$vertex connected chordal graph that serves as a sparse abstraction of the core network in which the maximum flow between every pair of vertices u and v in the abstraction is at most the maximum flow between u and v in the core network. If the graph is undirected it can be converted to a directed graph by replacing each edge by a pair of oppositely oriented edges. Section 5 concludes the paper. For graph theoretic terminology we refer to Thulasiraman and Swamy [17]. We require the following definitions and notations:

Let $G = (V, E, c)$ be a graph where $c : E \to \mathbb{R}$ be the capacities defined on each edge of G. Let $|V| = n$ and $|E| = m$. Let T be a spanning tree of G. For any two vertices $x, y \in V(T)$ denote $P(x, y) =< x = x_1, x_2, x_3, \ldots, x_k = y >$ to be a unique path of length k from x to y.

Definition I.1. *Let $G = (V, E)$ be a connected graph. If the graph is undirected it can be converted to a directed graph by replacing each edge by a pair of oppositely oriented edges. For each directed edge $e = (i, j) \in E$ define a non-negative real number called the capacity of the edge denoted by $c(e)$ or $c(i, j)$. If there is no edge e directed from i to j, then we define $c(e) = 0$.*

Definition I.2. *Given a graph $G = (V, E)$ with two distinct vertices s and t called the source and sink, an $s-t$ flow f in G is an assignment of a non- negative real number $f(e) = f(i, j)$ to each directed edge $e = (i, j)$ such that the following conditions are satisfied:*

1. *$f(i, j) \leq c(i, j)$ for every edge (i, j) in G.*
2. *$\sum_{all\ j} f(i, j) = \sum_{all\ j} f(j, i)$ for all $i \neq s, t$*

The value of an $s - t$ flow f between vertices s and t denoted by $f_G(s, t)$ is defined as

$$f_G(s, t) = \sum_{all\ u} f(s, u) = \sum_{all\ u} f(u, t)$$

An $s - t$ flow f between s and t in G is said to be a maximum $s - t$ flow if there exists no other flow f^* between s and t in G such that $f^*_G(s, t) > f_G(s, t)$.

The maximum flow between each pair of vertices can be represented as a $(n \times n)$ flow matrix, denoted by F_G. For example, consider the graph in Figure 1. The corresponding flow matrix is given in Figure 2. Now, for any pair of vertices u, v in G, the corresponding matrix entry $f(u, v)$ in F_G gives the maximum flow between u and v in G.

Definition I.3. *Let $G = (V, E)$ be a connected, undirected and simple graph, with vertex set V and edge set E. Let $|V| = n$ and $|E| = m$. Consider a partition $(S, \overline{S})$ of V, where $\overline{S} = V - S$. Then the set of edges with one end in S and the other in $\overline{S}$ is called a **cut** of G. A cut $\langle S, \overline{S}\rangle$ in a graph G is said to separate two vertices u and v if $u \in S$ and $v \in \overline{S}$. Such a cut is referred to as a $u - v$ cut. The capacity $c(K) = c(\langle S, \overline{S}\rangle)$ of a cut $K = \langle S, \overline{S}\rangle$ is defined as $c(K) = \sum_{i \in S, j \in \overline{S}} c(i, j)$.*

Note that if the edges are directed, then the capacities of edges directed from $\overline{S}$ to S do not contribute to the capacity of the cut $K = \langle S, \overline{S}\rangle$. If the edges are not directed, then the capacities of every edge between S and $\overline{S}$ contribute to the capacity of the cut $K = \langle S, \overline{S}\rangle$.

Definition I.4. *A $u-v$ cut K in a graph G is a minimum $u - v$ cut if there is no $u - v$ cut K' in G such that $c(K') < c(K)$.*

Definition I.5. *A graph G is said to be chordal if every cycle of length greater than three has a chord.*

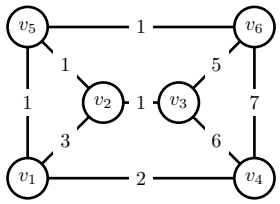

Fig. 1. A graph G.

$$F_G = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \left[\begin{array}{cccccc} 0 & 3 & 1 & 4 & 3 & 4 \\ 3 & 0 & 4 & 12 & 4 & 13 \\ 1 & 4 & 0 & 4 & 5 & 4 \\ 4 & 12 & 4 & 0 & 4 & 12 \\ 3 & 4 & 5 & 4 & 0 & 4 \\ 4 & 13 & 4 & 12 & 4 & 0 \end{array}\right] \end{array}$$

Fig. 2. The corresponding flow matrix F_G of the graph G.

Definition I.6. *The connectivity(or vertex connectivity), $\kappa(G)$, of a graph G is the minimum number of vertices whose removal from G results in a disconnected graph or a trivial graph.*

Definition I.7. *The edge connectivity, $\kappa'(G)$, of a graph G is the minimum number of edges whose removal from G results in a disconnected graph or a trivial graph.*

Definition I.8. *Let $T = (V, E)$ be a rooted tree with $|V| = n, |E| = n - 1$. An $r-$tree is a tree with root r. The level of a vertex v in T, denoted by $l(v)$, is the length of the path $< r, v >$ in T.*

The following celebrated theorem [17] is the basis of most works in network flow theory.

Theorem I.9. *The value of a maximum $s-t$ flow equals the capacity of the minimum $s-t$ cut.*

Throughout the paper the terms vertex and nodes are used interchangeably.

II. Preliminaries

In this section, first we review the concept of Gomory-Hu trees(G-H tree), their properties and the significance of using G-H tree trees in the paper. Gomory and Hu [18] in 1961 showed that the edge connectivity among all pairs of vertices in an undirected weighted/unweighted graph can be computed using $(n-1)$ max-flow computations rather than $\binom{n}{2}$ computations. For any given graph $G = (V, E)$, the G-H tree algorithm computes a weighted cut tree $T = (V, E')$, with the property that, for any pair of vertices $u, v \in V$, the maximum flow between u and v in G equals the weight of the minimum weighted edge on the unique path from u to v in T. It must be noted that the edges of a G-H tree may not be the edges of the graph G. That is, the weighted tree T need not necessarily be a subgraph of the graph G. As an example, the graph shown in Figure.3, is a G-H tree corresponding to the graph given in Figure.1. Note that

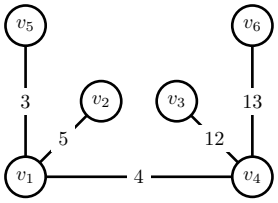

Fig. 3. A Gomory-Hu tree T of the graph G.

the capacity of an edge (u, v) in the G-H tree is the same as the capacity of the minimum $u - v$ cut in G and hence the value of the maximum $u - v$ flow in G. See [9] for a detailed discussion of algorithms for G-H tree construction. The rest of this section is devoted to a review on the fundamental cutsets in a graph. We adopt the terminologies and concepts from [17].

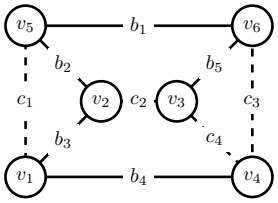

Fig. 4. A graph with a spanning tree (bold lines)

Consider the graph G in Fig.4 along with its spanning tree T. The bold edges denote the branches of a spanning tree T in G and the dotted edges denote the chords of T. If we remove a branch b from the spanning tree T, then the tree T gets disconnected resulting in two sub-trees T_1 and T_2. The sets of nodes in T_1 and T_2 define a partition of V. The corresponding cut is called the **fundamental cutset** of T with respect to the branch b. For example, if we remove the branch b_3 from the tree T of Fig. 4, then we get the sub-trees T_1 and T_2 given by the branches $\{b_1, b_2, b_5\}$ and $\{b_4\}$, respectively. The corresponding fundamental cutset $Q(b_3)$ consists of the edges $\{b_3, c_1, c_3, c_4\}$. Note that the subgraphs induced by the vertex sets of T_1 and T_2 are both connected.
The **fundamental cutset matrix** with respect to the tree T is defined as $Q^f = [q_{ij}]_{(n-1)\times(m)}$. The matrix Q^f has $(n-1)$ rows, one for each fundamental cutset and one column for each edge. The entry q_{ij} is defined as

$$q_{ij} = \begin{cases} 1, & \text{if } Q(b_i) \text{ contains edge } j \\ 0, & \text{otherwise.} \end{cases}$$

Arranging the rows of Q^f such that the j^{th} row corresponds to f-cutset $Q(b_j)$ and the columns correspond to the edges in the order $\{b_1, b_2, \ldots, b_{n-1}, c_1, c_2, \ldots, c_{m-n+1}\}$ the matrix Q^f can be written as $Q^f = [U|Q^{fc}]$. For example, the Q^f matrix with respect to the tree T of Fig. 4 is given below.

$$Q^f = \begin{array}{c} \\ Q(b_1) \\ Q(b_2) \\ Q(b_3) \\ Q(b_4) \\ Q(b_5) \end{array} \left[\begin{array}{ccccc|cccc} b_1 & b_2 & b_3 & b_4 & b_5 & c_1 & c_2 & c_3 & c_4 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

From the fundamental cutset matrix it is easy to capture the cutsets to which a given chord belongs to. For example, the chord c_3 belongs to the fundamental cutsets $\{Q(b_1), Q(b_2), Q(b_3), Q(b_4)\}$.

III. Capacity Assignment using Fundamental Cutsets

Given the flow matrix F of a core network, let T be the corresponding G-H tree. Let $G = (V, E)$ be a graph with the same vertex set as that of the core network and the spanning tree T. Note that the edge set E of G need not be the same as that of the core network. However, the spanning tree T of the core network is also the spanning tree of the graph G. We now present an algorithm that assigns capacities to the edges of G such that the maximum flow between any pair of vertices u, v in G is at most the value $f(u, v)$ of a maximum $u - v$ flow in the core network. As before let Q^f be the fundamental cut-set matrix of G with respect to T.

Algorithm : Fundamental Cutset Based Capacity Assignment Algorithm (FCS-BCA)

Input: Flow matrix F of the core network, a graph $G = (V, E)$ which has the G-H tree T as a spanning tree and Q^f the fundamental cut-set matrix of G with respect to T of the core network.

Output: A graph $G_T = (V, E, c_T)$, where c_T denotes the newly assigned capacities to the edges in G_T.

FCS-BCA(G, T, F_G, Q^f)

1. Let G_T be a copy of the graph G where for each $e \in E(G_T), c_T(e) = \infty$.
2. For each edge $e \in G_T$ do
   3. For each j, such that $e \in Q(b_j)$ do
      4. $c_T(e) = min\{\frac{f(b_j)}{|Q(b_j)|}, c_T(e)\}$, where $f(b_j) = f(u, v)$, the value of the maximum flow between the vertices of b_j.
5. return (G_T, c_T)

Complexity: The outer for-loop at line 2 runs for each edge e of G and hence takes $O(m)$. The nested for loop at line 3 runs for each fundamental cut set $Q(b_i)$ that contains the edge e. This step takes at most $O(n)$ time. Overall, the two for loops take $O(n.m)$.

Note that the graph G_T output by the algorithm is such that the capacities are re-assigned based on the capacities of the spanning tree T that is considered. We now have the following proposition.

Proposition III.1. *Let b_i be a branch in the spanning tree T and let $Q(b_i)$ be the corresponding fundamental cut set of b_i in G_T. Then, for each edge $e \in Q(b_i)$, $c_T(e) \leq c_T(b_i)$.*

Proof. If e is the branch b_i, then clearly b_i does not belong to any other fundamental cut-set and hence, $c_T(b_i) = min\{\frac{f(b_i)}{|Q(b_i)|}, c_T(b_i)\} = \frac{f(b_i)}{|Q(b_i)|}$. If e is a chord, then e belongs to at least one or more fundamental cut-sets. Let $b_{l_1}, b_{l_2}, \ldots, b_{l_k}$ be the branches such that $e \in Q(b_{l_j}), 1 \leq j \leq k$, where b_i is one of these branches. Then, by steps 3 and 4 of the algorithm we have,

$$\begin{aligned} c_T(e) &= min\{\frac{f(b_{l_1})}{|Q(b_{l_1})|}, \ldots, \frac{f(b_{l_k})}{|Q(b_{l_k})|}, c_T(e)\} \\ &\leq \frac{f(b_i)}{|Q(b_i)|} \\ &= c_T(b_i). \end{aligned}$$

□

Note that the above algorithm guarantees that the maximum flow between any two adjacent vertices u and v with respect to the spanning tree T is distributed among all the edges of each fundamental cut-set between u and v in T. Now we have the following theorem.

Theorem III.2. *Let F be the flow matrix of the core network and T be the corresponding G-H tree. Let $G = (V, E)$ be a graph which has the same vertex set V as the core network. Assume that the capacities of the edges of G are assigned as in algorithm FCS-BCA. Then, $f_G(u, v) \leq f(u, v)$ for all $u, v \in V$, where $f_G(u, v)$ is the value of a maximum $u - v$ flow in G and $f(u, v)$ is the value of a maximum $u - v$ flow in the core network.*

Proof. Consider any branch b_i of T. Let $Q(b_i)$ be the

fundamental cut-set of b_i and let $e \in Q(b_i)$. Then,

$$\begin{aligned} c_T(Q(b_i)) &= \sum_{e \in Q(b_i)} c_T(e) \\ &\leq \sum_{e \in Q(b_i)} \frac{f(b_i)}{|Q(b_i)|} \\ &= |Q(b_i)|.\frac{f(b_i)}{|Q(b_i)|} \\ &= f(b_i) \end{aligned}$$

Thus if $b_i = (u, v)$ then,

$$\begin{aligned} f_G(u, v) &\leq c_T(Q(b_i)) \\ &\leq f(b_i) \\ &= f(u, v) \end{aligned}$$

Consider any two vertices x and y that are not adjacent in T. Let $b_i = (u, v)$ be the branch with the smallest capacity on the $x-y$ path in T. Then G-H tree guarantees that $f(x, y) = f(u, v)$. Since the cut $Q(b_i)$ that separates u, v also separates x, y, we have,

$$\begin{aligned} f_G(x, y) &\leq f_G(u, v) \\ &\leq f(u, v) \\ &= f(x, y) \end{aligned}$$

Thus the result of the theorem follows in all cases. □

IV. Algorithms for Virtual Topology Construction

In this section we give two algorithms. The first algorithm is a basic algorithm that constructs a 2-edge connected graph from a given G-H tree T. We then make a modification of the basic algorithm to construct a 2-edge connected and 2-vertex connected chordal graph from the G-H tree T. We begin with the basic algorithm.

A. Algorithm for a 2-Edge Connected Graph

A vertex that is equidistant from all vertices of T is called a center of T. A tree will have exactly one or exactly two centers. Let $r \in T$ be the center of T (we arbitrarily choose one vertex, if there are two centers for T). Let T be rooted at r and let r be assigned level 0. Subsequently, let each vertex be assigned a level and denote the level of a vertex v_i by $l(v_i)$. We label the leaf vertices of T by an application of the DFS algorithm, such that, associated with each vertex v, the child vertices of v are examined in decreasing order of their degree. Now let $u_1, u_2, u_3, \ldots, u_l$ be the labels assigned by the DFS algorithm to the leaf vertices of T. For each leaf vertex $u_i \in T$, let $P(u_i, r)$ be the unique path from u_i to r.

We now give the basic algorithm to construct a 2-edge connected graph $G = (V, E, c)$ with vertex set V, and edge set $E = E^T \cup BE$, where BE(call them Back Edges) are the new edges added to the Gomory-Hu tree T and $c : E \to \mathbb{R}$ denote the updated capacities of the edges in E.

We require the following subroutine that adds back edges to a path $P(x, y)$:

Add Back Edge(P(x,y))

1. Let $j = 1$;
2. Let $P(x, y) =< x = v_1, v_2, v_3, \ldots, v_k = y >$;
3. while $(j < k - 1)$ do
   3.1 Add edge (v_j, v_{j+2});
   3.2 Let $c(v_j, v_{j+2}) = 0$;
   3.2 $j \leftarrow j + 2$;
4. If k is even then add the edge (v_{k-2}, v_k) and let $c(v_{k-2}, v_k) = 0$;

Algorithm 1: Basic 2-edge connected graph

Input: A G-H tree T and flow matrix F of the core network.

Output: A 2-edge connected graph $G = (V, E^T, c_T)$ which has T as a spanning tree. Note: Capacity $c_T(e)$ of edge $e \in T$ is the same as $f(u, v)$, where u and v are the end vertices of e.

1. Let $u_1, u_2, \ldots, u_l$ be the leaf vertices in the tree T.
2. Add Back Edge($P(u_1, r)$);
3. Let i=2;
4. while($i \leq l$) do
   4.1 On $P(u_i, r)$ let w be the first vertex that intersects $P(u_j, r)$ for some $j, 1 \leq j < i$.[Observe that w may be equal to r.]
   4.2 If $(|P(u_i, w)| > 1)$ then
   {
   Add Back Edge($P(u_i, w)$);

   $i \leftarrow i + 1$;
   }
   4.3 else [Note: $|P(u_i, w)| = 1$]
      4.3.1 If u_i is the only pendant of w then
      {
      Let $r_1 \in N(w)$, such that $c_T(w, r_1) = min\{c_T(w, x) : x \in N(w) \backslash u_i\}$;
      Add edge (u_i, r_1);
      Let $i \leftarrow i + 1$;
      }
      4.3.2 else [Note: (w ***has more than one pendant vertex***)]

{
Let $(u_i, u_{i+1}, u_{i+2}, \ldots, u_{i+s})$ be the pendant vertices of w;
Add edges $(u_i, u_{i+1})(u_{i+1}, u_{i+2}), \ldots, (u_{i+s-1}, u_{i+s})$;
Let $i \leftarrow i + s + 1$;
}

5. $(G, c) \leftarrow$ FCS-BCA(G, T, F, Q^f);
6. Output the graph G with updated edge capacities.

Theorem IV.1. *The graph G output by algorithm IV-A is 2-edge connected.*

Proof. For each $u_i \in T$, the back edges in the path $P(u_i, w)$ with $|P(u_i, w)| > 1$, connect vertices at alternate levels starting from the leaf. This operation triangulates the path $P(u_i, w)$, where every edge lies in a triangle. When $|P(u_i, w)| = 1$, we either add an edge (u_i, r_1) where the u_i, r_1, w form a triangle, or, we add edges, $(u_i, u_{i+1})(u_{i+1}, u_{i+2}), \ldots, (u_{i+s-1}, u_{i+s})$, which induces a diamond centered at the node w. □

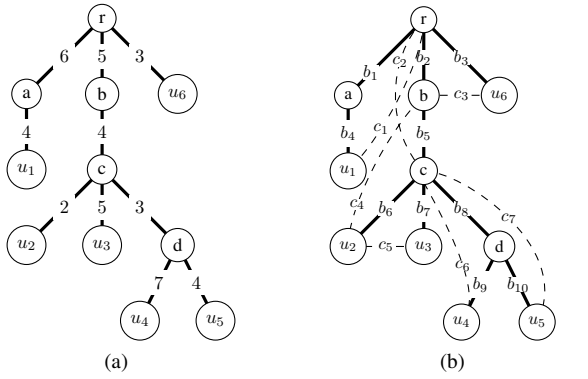

Fig. 5. (a)A G-H tree T with capacities on each of the edges denoting the maximum flow in the core network (b) The graph output by Algorithm IV-A. Here the dashed edges labeled $c_1, c_2, \ldots, c_7$, denote the chords(back edges) added by the algorithm and the edges labeled $b_1, b_2, \ldots, b_{10}$ are the branches of the G-H tree T. The capacities of the chords are initially assigned zero. The ***FCS-BCA*** algorithm re-assigns non-negative capacities to the chords as in Fig. 6.

As an example, the graph G in Fig. 5b is output by Algorithm-IV-A based on the G-H tree T given in Fig. 5a. Here T is a G-H tree of some core network. Fig. 6 shows the resulting graph with the re-assigned capacities to the branches and the chords after an application of the ***FCS-BCA*** Algorithm.

B. Algorithm for 2-Edge Connected, 2-Vertex Connected, Chordal Graph

In this section we give an algorithm that constructs a 2-edge connected, 2-vertex connected chordal graph

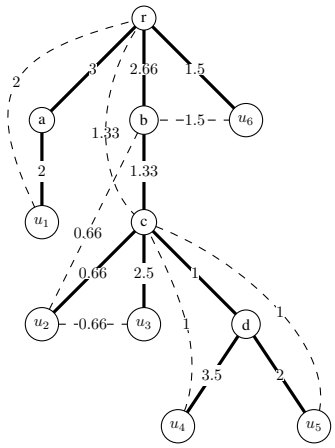

Fig. 6. For the resulting graph in Fig. 5b by an application of Algorithm IV-A, the ***FCS-BCA*** algorithm re-assigns the capacities to each of the branches and chords. These re-assigned capacities for the respective branches and chords shown in the graph given above.

from the G-H tree T. We use the same notation from the basic algorithm. We require the subroutine Add Back Edge Chordal that adds back edges to every vertex in the path $P(x, y)$:

Add Back Edge Chordal(P(x,y))

1. Let $j = 1$;
2. Let $P(x, y) =< x = v_1, v_2, v_3, \ldots, v_k = y >$;
3. while $(j < k - 1)$ do
   3.1 Add edge (v_j, v_{j+2});
   3.2 Let $c(v_j, v_{j+2}) = 0$;
   3.3 $j \leftarrow j + 1$;
4. Add the edge $(v_{k-1}, p(v_k))$ and let $c(v_{k-1}, p(v_k)) = 0$;

Algorithm 2: 2-edge connected 2-vertex connected chordal graph

Input: A G-H tree $T = (V, E^T, c_T)$.

Output: A 2-edge connected 2-vertex connected chordal graph $G = (V, E)$ which has T as a spanning tree. Note: Capacity $c_T(e)$ of edge $e \in T$ is the same as $f(u, v)$, where u and v are the end vertices of e.

1. Let $w_1, w_2, w_3, \ldots, w_k$ be the child nodes of the root r in the tree T. Add edges $(w_i, w_{i+1}), 1 \leq i \leq k-1$, and make $c(w_i, w_{i+1}) = 0$. Let $u_1, u_2, \ldots, u_l$ be the leaf vertices in the tree T.
2. Add Back Edge Chordal($P(u_1, r)$);
3. Let i=2;
4. while($i \leq l$) do
   4.1 On $P(u_i, r)$ let w be the first vertex that intersects $P(u_j, r)$ for some $j, 1 \leq j < i$.
   4.2 Add Back Edge Chordal($P(u_i, w)$);

4.3 $i = i + 1$;

5. $(G, c) \leftarrow$FCS-BCA(G, T, F_G, Q^f);
6. Output the graph G with updated edge capacities.

Next we show that the graph G obtained using Algorithm IV-B is 2-edge connected, 2-vertex connected and chordal.

Theorem IV.2. *The graph G output by algorithm IV-B satisfies the following:*

1. *G is 2-edge connected.*
2. *G is 2-vertex connected.*
3. *G is chordal.*

Proof. Let $(u, v) \in E(G)$ be any edge. Without loss of generality assume that $l(u) < l(v)$. Then, $(v, p(u)) \in G$ and $(u, w) \in G$ where $w \in N(V)$ with $l(w) > l(v)$. then, the removal of the edge (u, v) clearly does not disconnect the graph G since v is connected to u through $p(u)$. Hence G is 2-edge connected. Similarly, let u be any vertex in G with $l(u) > 1$ and let $(u, v) \in E(G)$. Again, without any loss of generality assume that $l(u) < l(v)$. Then the induced subgraph, $< v, u, p(u) >$ is a triangle in G so that the removal of u will not disconnect G. If $l(u) = 1$, then u is the root r and let $r_1, r_2, \ldots, r_{|N(r)|}$ be the neighbors of r at level 1. By construction, the induced subgraph $< r, r_1, r_2, \ldots, r_{|N(r)|} >$ is a diamond graph and it is easy to see that the removal of r does not disconnect the induced subgraph $< r, r_1, r_2, \ldots, r_{|N(r)|} >$. Hence G is 2-vertex connected. To show that G is chordal, we give a procedure to obtain a perfect elimination ordering of all the vertices in G. Let $l(v)$ denote the level of a vertex in G and let $S = \phi$ be an empty sequence. We will add all the vertices in $V(G)$ to S so that S is a perfect elimination ordering of vertices in G. Let k be the highest level assigned to the vertices and let $i = k$. We begin from level k and go down upto level 2. At each level i, pick a vertex v, add it to the sequence S and delete the vertex v and all edges incident on it. We do this until no vertex remains at level i. Repeat this process at each level until level 1 is reached. The remaining vertices in G are the vertices r and its neighbours at level 1. If $r_1, r_2, \ldots, r_{|N(r)|}$ are the vertices adjacent to r at level 1, then add the vertices $r_1, r_2, \ldots, r_{|N(r)|}, r$ in that order to the sequence S. We now claim that the sequence of vertices in S is a perfect elimination ordering. To see this let v be a vertex such that $l(v) = i, 2 \leq i \leq k$. Then v is either adjacent to a vertex u at level $i - 1$ or adjacent to two vertices u and $p(u)$ with $l(u) = i - 1$, where $p(u)$ denotes the parent vertex of u at level $i - 2$. In either case, the induced subgraph of $< v, N(v) >$ is either a K_2 or a K_3 with v being a simplicial vertex. Hence v can be added to the sequence S. Since this is true for each vertex at level i, every vertex at level i enters S. Further, after level 1 is reached, the remaining vertices are the vertices r and its neighbors at level 1. Let $r_1, r_2, \ldots, r_{|N(r)|}$ be the neighbors of r at level 1. By construction, the induced subgraph $< r, r_1, r_2, \ldots, r_{|N(r)|} >$ is a diamond. Starting from r_1 up to $r_{|N(r)|}$, we observe that each r_i is a simplicial vertex in the induced subgraph $< r_i, N(r_i) >$ after the removal of the vertex r_{i-1}. Hence, each r_i enters the sequence S and the root vertex can be added to S at the end so that S is a perfect elimination ordering. Since S contains all the vertices of G, we can conclude that G is a chordal graph. □

As an example, the graph in Fig.7 is output by Algorithm IV-B based on the G-H tree T given in Fig. 5a. Again, the tree T is a G-H tree of some core network. The capacities re-assigned to the branches and the chords of the graph in Fig. 7a after an application of the ***FCS-BCA*** Algorithm is shown in Fig. 8.

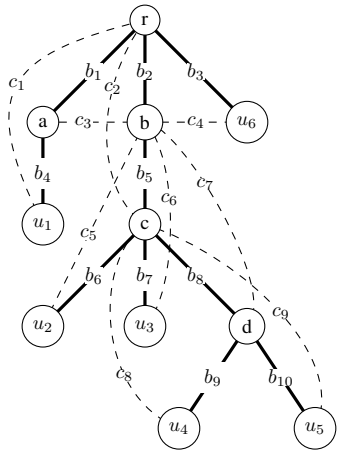

Fig. 7. For the G-H tree T given in Fig. IV-A, the graph given above is output by Algorithm IV-B. Here the dashed edges labeled $c_1, c_2, \ldots, c_9$, are the chords and the edges $b_1, b_2, \ldots, b_{10}$ are the branches of the G-H tree T. The capacities of the back edges are initially assigned zero. The ***FCS-BCA*** algorithm re-assigns non-negative capacities to these edges as in Fig. 8

Note that both the algorithms call the subroutine **FCS-BCA()**. From Theorem III.2 we have that $f_G(u, v) \leq f(u, v)$ for each pair of vertices u, v in T. Since, the tree T considered in both the algorithms in section IV-A and IV-B is a G-H tree, T captures the maximum flow between all pairs of vertices in the original graph G. Hence, the capacity assignment based on the algorithm

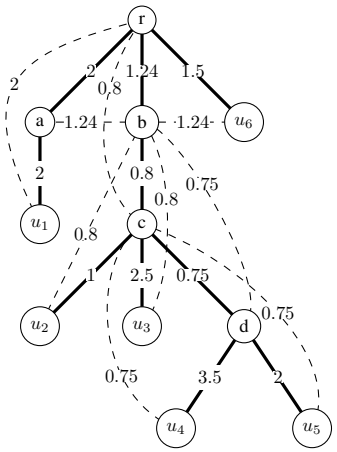

Fig. 8. For the resulting graph in Fig. 7 by an application of Algorithm IV-B, the ***FCS-BCA*** algorithm re-assigns the capacities to each of the branches and chords. These re-assigned capacities for the respective branches and chords shown in the graph given above.

in section III leads us to conclude the following theorem. The proof follows directly from Theorem III.2.

Theorem IV.3. *For any pair of vertices $u, v \in V$, we have $f_G(u, v) \leq f_T(u, v) = f(u, v)$.*

V. Conclusion

In this paper we have considered the problem of constructing virtual topologies useful in providing topology abstraction service for IP-VPNs [2] and other two-layer applications. Given the node-to node maximum flow matrix of a core network with vertex set V and also given the corresponding G-H tree T[18]. For any connected graph $G = (V^*, E)$ with V^* subset of V, we show how to assign capacities to the edges of G such that the maximum flow between any pair of vertices u and v in G is no more than the maximum flow between u and v in the core network. Though this methodology is applicable to any graph, we seek virtual topologies with certain other desirable properties: low degree, number of edges is linear in the number of vertices, amenable to survivable logical topology routing in an IP over WDM optical network etc. Chordal graphs have such properties. In particular chordal graphs admit survivable logical topology routing as long as the core network is $3-$edge connected [8]. In view of this we show how to augment any tree T with additional edges to make it into a $2-$verex and $2-$edge connected chordal graph. Combining these results, our main contribution is how to construct chordal graphs and assign capacities to the edges of the graph that capture the node to node flow information in a core network. There is a rich theory available for the graph augmentation problem [10]. These augmentation results along with the theory and methodologies developed in this paper provide the framework for generating other classes of virtual topologies for the topology abstraction service problem.

Acknowledgment

We thank....

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] Ravi Ravindran, ChangCheng Huan, K. Thulasiraman, *Topology Abstraction as VPN Service*, Proceedings of IEEE ICC, (2005), pp. 105–109.

[3] Ravi Ravindran, ChangCheng Huan, K. Thulasiraman, *A Dynamic Managed VPN Service: Capacity Sharing Based On Maximum Concurrent Flow Theory*, Proceedings of IEEE ICC, Vol. 2, (2007), pp. 211–216.

[4] Ravi Ravindran, Peter Ashwood-Smith et. al., *Multiple Abstractions Schemes for Generalized Virtual Private Networks*, Proceedings of IEEE CCECE, (2004), Niagara.

[5] Ravishankar Ravindran, Changcheng Huang and Krishnaiya Thulasiraman, *Topology Abstraction Service for IP-VPNs*, IEEE Transactions on Parallel and Distributed Systems, Vol. 24, January 2013, pp, 184-197.

[6] Ravishankar Ravindran, Changcheng Huang and Krishnaiya Thulasiraman, *Topology Abstraction Service for IP-VPNs: Core Network Partitioning for Resource Sharing*, *Under Review*

[7] K. Thulasiraman, S. Arumugam, A. Brandstadt and T. Nishizeki, *Handbook of Graph Theory, Combinatorial Optimization and Algorithms*, CRC Press, 2015, Chapter 13.

[8] K. Thulasiraman, T. Lin, M. Javed, and G. Xue, *Logical topology augmentation for guaranteed survivability under multiple failures in IP-over-WDM optical networks*, IEEE ANTS 2009, Dec. 2009.

[9] K. Thulasiraman, *Connectivity Algorithms* , Handbook of Graph Theory, Combinatorial Optimization and Algorithms, CRC Press, 2015, Chapter 13.

[10] A. Frank, T. Jordan *Graph Connectivity Augmentation* , Handbook of Graph Theory, Combinatorial Optimization and Algorithms, CRC Press, 2015, Chapter 14.

[11] Ravi Ravindran, ChangCheng Huan, K. Thulasiraman, *VPN Topology Abstraction Service using Centralized Core Capacity Sharing Scheme*, IEEE ANTS,(Dec. 2007).

[12] W Lee, *Topology Aggregation for hierarchical routing in ATM networks*, ACM SIGGCOMM Computer Commun. Rev., Vol. 25, (April 1995), pp. 82–92.

[13] W Lee, *Spanning Tree method for Link State Aggregation in large Communication Networks*, Proc. of IEEE INFOCOM, (1995), pp. 82–90.

[14] Turgay Korkmaz, Marwan Krunz, *Source-Oriented Topology Aggregation with Multiple QoS parameters in Hierarchical Networks*, ACM Transactions on Modeling and Computer Simulations, Vol. 10, No. 4, (Oct. 2000), pp. 295–325.

[15] King-Shan Li, Kalara Nahrstedt, Shigang Chen, *Ruting with Topology Aggregation in Delay-Bandwidth Sensitive Networks*, IEEE/ACM Transactions on Networking, Vol. 12, No. 1, (Feb. 2004).

[16] Yong Tang, Shigang Chen, *QoS Information Approximation for Aggregated Networks*, IEEE ICC, (2005).

[17] M.N.S.Swamy and K. Thulasiraman, *Graphs, Networks and Algorithms*, Wiley Interscience, 1981.

[18] R. E. Gomory and T. C. Hu, *Multi-terminal network flows*, J. Soc. Indust. Appl. Math., 9(4), (1961), pp. 551-570.