Provisioning a Virtual Private Network: A Network Design Problem for Multicommodity Flow

Anupam Gupta [*] Jon Kleinberg [†] Amit Kumar [‡] Rajeev Rastogi [§]
Bulent Yener [¶]

Abstract

Consider a setting in which a group of nodes, situated in a large underlying network, wishes to reserve bandwidth on which to support communication. *Virtual private networks* (VPNs) are services that support such a construct; rather than building a new physical network on the group of nodes that must be connected, bandwidth in the underlying network is reserved for communication within the group, forming a virtual "sub-network."

Provisioning a virtual private network over a set of terminals gives rise to the following general network design problem. We have bounds on the cumulative amount of traffic each terminal can send and receive; we must choose a path for each pair of terminals, and a bandwidth allocation for each edge of the network, so that *any* traffic matrix consistent with the given upper bounds can be feasibly routed. Thus, we are seeking to design a network that can support a continuum of possible traffic scenarios.

We provide optimal and approximate algorithms for several variants of this problem, depending on whether the traffic matrix is required to be symmetric, and on whether the designed network is required to be a tree (a natural constraint in a number of basic applications). We also establish a relation between this collection of network design problems and a variant of the facility location problem introduced by Karger and Minkoff; we extend their results by providing a stronger approximation algorithm for this latter problem.

1 Introduction

Consider a setting in which a group of nodes, situated in a large underlying network, wishes to reserve bandwidth on which to support communication. *Virtual private networks* (VPNs) are services that support such a construct; rather than building a new physical network on the group of

[*]Department of Computer Science, Cornell University, Ithaca NY 14853. Email: anupam@cs.cornell.edu. Supported by NSF grants CCR-9700029 and DMS-9805602, and ONR grant N00014-98-1-0589.

[†]Department of Computer Science, Cornell University, Ithaca NY 14853. Email: kleinber@cs.cornell.edu. Supported in part by a David and Lucile Packard Foundation Fellowship, an ONR Young Investigator Award, and NSF Faculty Early Career Development Award CCR-9701399.

[‡]Department of Computer Science, Cornell University, Ithaca NY 14853. Email: amitk@cs.cornell.edu. Supported in part by Lucent Bell Labs and the ONR Young Investigator Award of Jon Kleinberg. Part of this work was done while visiting Lucent Bell Labs.

[§]Bell Labs, 600 Mountain Avenue, Murray Hill NJ 07974. Email: rastogi@bell-labs.com.

[¶]Bell Labs, 600 Mountain Avenue, Murray Hill NJ 07974. Email: yener@bell-labs.com.

nodes that must be connected, bandwidth in the underlying network is reserved for communication within the group, forming a virtual "sub-network" [7, 8, 21].

Setting up such a virtual private network gives rise to a collection of basic optimization problems. Since bandwidth must be reserved at a cost, we would like to reserve as little as necessary to support the expected communication. At the same time, the VPN must be flexible enough to support a range of possible communication patterns among the nodes it serves; in other words, a single *traffic matrix* is not known in advance, since communication patterns will be changing over time. Finally, it is often desirable for the subgraph on which bandwidth is reserved to have a simple structure that will facilitate routing in the resulting VPN. Thus, we are essentially dealing with a network design problem for multicommodity flow in which the node-to-node demands are not known with certainty at the outset.

Duffield et al. [7] recently proposed a natural model for expressing VPNs that allows a large degree of flexibility in formulating possible traffic patterns. We are given an undirected graph $G = (V, E)$ (the underlying network) with a cost c_e on each edge; and we are given a set of *terminals* $W \subseteq V$ that wish to establish communication. In the simplest form of the model (the *symmetric case*) each terminal $i \in W$ has an upper bound $b(i)$ on the cumulative amount of traffic that can be sent or received by i at any point in time; we must choose bandwidth reservations and node-to-node paths so that *any* set of demands respecting these upper bounds can be feasibly routed. More formally, a *valid traffic matrix* $\vec{d}$ on W is an assignment of a demand $d_{ij} \geq 0$ to each unordered pair i, j of terminals that respects the cumulative upper bounds $\{b(i)\}$: for each i, we have $\sum_j d_{ij} \leq b(i)$. A virtual private network for W is then provisioned by selecting a path P_{ij} for each unordered pair of terminals $i, j \in W$, and reserving some (real-valued) amount of bandwidth $x_e \geq 0$ on each edge e. We must choose $\{P_{ij}\}$ and $\{x_e\}$ in such a way that in the graph G with edge capacities $\{x_e\}$, the demands corresponding to *any* valid traffic matrix $\vec{d}$ can be routed along the paths $\{P_{ij}\}$ — in other words, for every valid $\vec{d}$, we have $\sum_{ij:e \in P_{ij}} d_{ij} \leq x_e$. The goal is to minimize the total cost of the reserved bandwidth, $\sum_e c_e x_e$.

For example, in Figure 1, we have two possible sets of paths $\{P_{ij}\}$ for a symmetric instance with three terminals labeled 1, 2, and 3. Suppose that $b(i) = 1$ for each i, and each edge cost is equal to 1. In the solution on the left, each edge is in either 0 or 2 paths; the crucial point is that for edges e in the latter category, we need only set $x_e = 1$. Indeed, although such an edge e appears in two paths, the values $\{b(i)\}$ imply that e will never be required to carry more than one unit of demand under any valid traffic matrix. Thus, there is a solution of cost 7 using the paths on the left. Using the paths on the right, on the other hand, we must still reserve $x_e = 1$ for each edge e appearing in any path, since there is some valid traffic matrix in which it must carry one unit of flow. Hence we now incur a cost of 11; this higher cost is due to poorer multiplexing of the paths.

Our problem thus has both discrete and continuous aspects. We may reserve an arbitrary real-valued amount of bandwidth on each edge, but must specify a fixed path P_{ij} in advance on which to route the i-j flow under any traffic matrix. The requirement that each i-j flow be routed on a fixed path P_{ij} is motivated by applications that require guarantees on delay and throughput; an important application of this type is voice over IP, where strong real-time requirements prevail. Finally, a fundamental feature of our problem is that the selection of paths and bandwidths must be adequate for a *continuum* of possible traffic scenarios — any matrix that respects the bounds $\{b(i)\}$.

For many of the key applications of this model, we need to impose the requirement that the

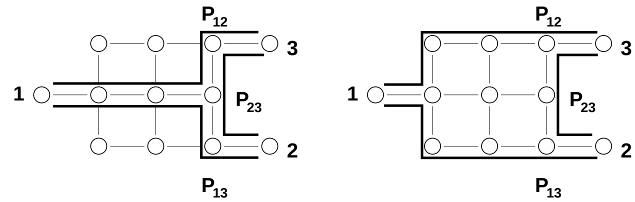

Figure 1: Two possible sets of paths for a 3-terminal symmetric instance.

union of the edges in the path set $\{P_{ij}\}$ should form a *tree*. This corresponds to the general notion raised earlier that the underlying virtual private network should be structurally simple enough to facilitate routing. A tree structure is crucial for scalability in representing and configuring the routes with respect to each terminal. Also, MPLS (MultiProtocol Label Switching), which is emerging as a standard for setting up paths between pairs of terminals, is considerably simplified when trees are used, since fewer labels have to be used, and label stacks on packets are not as deep [6]. Furthermore, trees simplify restoration of paths in case of link failures, since all paths traversing a failed link can be restored as a single group, instead of each path being restored separately. Thus we will be considering two versions of the symmetric model: one in which the reserved paths must form a tree, and a more general one in which they may have an arbitrary structure.

In addition to the symmetric model, there is a more general asymmetric formulation of the problem: flows are directed in the sense that the i-j flow is distinct from the j-i flow. Thus, each node now has two upper bounds, $(b_{in}(v), b_{out}(v))$, which represent upper bounds on the amount of flow that the terminal i can receive and send, respectively. A valid traffic matrix $\vec{d}$ on W is now an assignment of a demand $d_{ij} \geq 0$ to each *ordered pair* i, j of terminals that respects these cumulative upper bounds: for each i, we have $\sum_j d_{ij} \leq b_{out}(i)$ and $\sum_j d_{ji} \leq b_{in}(i)$. The remainder of the formulation is completely analogous: we must choose bandwidth reservations $\{x_e\}$ and paths $\{P_{ij}\}$ for each *ordered pair* of terminals (i, j); and these must be chosen so that the demands corresponding to any valid traffic matrix can be routed with capacities $\{x_e\}$ on the paths $\{P_{ij}\}$. Again, the goal is to minimize total cost.

One setting in which the asymmetric model naturally arises is that of terminals that are able to generate data at a much lower rate than they are able to receive it. This corresponds to the case in which, for each terminal i, $b_{in}(i)$ is so large that it exceeds the sum of $b_{out}(j)$ over all other j. We will refer to this as the *source-limited case*. For brevity, when $b_{in}(i) \geq \sum_{j \neq i} b_{out}(j)$, we will sometimes write $b_{in}(i) = \infty$.

Our results. We consider the four variants of the model discussed above: the flows may be symmetric or asymmetric, and the structure of the VPN may be a tree or a general subgraph. We denote these variants by the abbreviations $SymT$, $SymG$, $AsymT$, and $AsymG$; here the suffix T represents the case in which the solution must be a tree, and the suffix G represents the case in which the solution may be any subgraph.

We first show that there is polynomial-time algorithm to solve $SymT$ optimally. The problem $AsymT$, on the other hand, is NP-complete, and we give a constant-factor approximation algorithm. We obtain the approximation algorithm in two steps. First, by adapting and strengthening our analysis of the symmetric case, we reduce $AsymT$ to the following *Connected Facility Location Problem*:

- We are given a graph G with edge lengths, a set of *demand nodes* in the graph, and a set of *candidate facility nodes*; we must open a subset of the facilities, and assign each demand node to one of them. As in the traditional facility location problem, we pay a cost f_i for opening a facility at node i, and we pay a cost proportional to the demand-distance product for assigning demand node j to facility i. The new feature is that we incur an additional cost: we must also choose a Steiner tree T which connects all the open facilities, and we pay an additional cost proportional to the total edge length of T.

This problem was introduced by Karger and Minkoff [15], who developed an algorithm approximating the optimum to within a constant factor. We provide a significantly improved constant-factor approximation algorithm for the Connected Facility Location problem, adapting a rounding technique of Shmoys, Tardos, and Aardal [22]. Combined with our underlying reduction, this yields an approximation algorithm for $AsymT$. Although we arrived at the Connected Facility Location problem as a step in approximating network design, we feel it is a very natural problem in its own right — it essentially captures a setting in which facilities need to communicate easily with one other (e.g. for maintaining consistent data, or re-balancing physical supplies).

We next turn to the case in which the underlying VPN may be an arbitrary subgraph. We provide a 2-approximation algorithm for $SymG$, by showing that there is a tree solution whose cost is within a factor of two of an optimum that is allowed to use arbitrary subgraphs. We do not know of a good approximation algorithm for general instances of $AsymG$, and leave this as an open question. However, for the source-limited case, we obtain a constant-factor approximation algorithm by showing that there is always an optimal solution for source-limited instances in which the underlying set of paths forms a tree. As a result, an approximate solution can be obtained using algorithms for computing approximately optimal Steiner trees.

In approaching these problems, we have found it useful to consider the natural *fractional relaxations* $SymF$ and $AsymF$, in which we must reserve bandwidth $x_e \geq 0$ on each edge e so that in the graph G with capacities $\{x_e\}$, there is feasible fractional multicommodity flow for the set of demands corresponding to any valid traffic matrix. Essentially, this fractional version of the problem has the following interpretation: the bandwidth reservations must be determined in advance, but once we are given a traffic matrix, we may then choose a multicommodity flow that is feasible with respect to the reserved bandwidth; fractional flow paths for all pairs do not have to be fixed in advance of learning the traffic matrix. Although this does not correspond to the way in which VPNs are generally provisioned, this fractional model provides a useful bound for purposes of comparison. However, despite its close connection with multicommodity flow, there are cases in which this fractional relaxation is actually hard to solve optimally — in particular, we show it is co-NP hard with asymmetric traffic bounds on a *directed* graph.

Finally, we consider the case where the edges of G have capacities, and the allocated bandwidth x_e can be at most the capacity u_e. In this case we show that even checking feasibility of $SymT$ and $SymG$ is hard. We also give a bicriteria approximation algorithm for $SymF$ in this case.

Connections to Related Work. The connected facility location problem was first studied by Karger and Minkoff [15], who obtained a constant-factor approximation algorithm. Their approach works in two stages. The first stage decides which facilities to open by "clustering" demands as much as possible (see also the paper by Guha et. al. [12]). In the second stage, they prove that it does not cost much to connect these facilities by a minimum Steiner tree. Our algorithm formulates a single linear programming relaxation for the combined problem, leading to a significantly improved constant factor. The connected facility location problem is also related to other facility location variants, including multi-level and load-balanced facility location [1, 12], as well as to the problem of finding connected vertex covers [3].

The main network design problems considered here are related to algorithmic work on two other network design models, namely *survivable network design* and *buy-at-bulk network design*, but with some crucial differences.

In survivable network design (see, e.g., [10, 13, 23, 24]), we are given a graph G with integer-valued flow requirements f_{ij} between certain pairs of nodes i and j. We must choose a minimum-cost subgraph H of G so that for each pair i, j, there are at least f_{ij} edge-disjoint i-j paths in H. Thus, this is essentially a *single-commodity* network design problem; the subgraph constructed needs to satisfy the demand for any *single* pair of nodes, but not for multiple pairs simultaneously.

In buy-at-bulk network design (see, e.g., [2, 4, 20]), there is a specified demand d_{ij} between every pair of nodes in a graph G, and we must buy capacity on edges as cheaply as possible so as to support a (fractional or integral) multicommodity flow satisfying all the demand. Now, the demand here is assumed to be known exactly, so if costs were linear in the amount of capacity purchased the following simple approach would work: for each pair i, j separately, buy d_{ij} units of capacity on any shortest i-j path. The problem in the buy-at-bulk model is complicated by "economies of scale," which is reflected in the fact that the cost of capacity is a concave function of the amount purchased.

Thus, the main qualitative difference in the problems we consider is the notion of quantifying over possible traffic scenarios: we must provision a single network (with paths for routing) in advance, and it must support any traffic pattern that is consistent with certain initial guarantees provided by the terminals.

2 Tree Solutions

This section addresses the problem of designing the best tree solution in both the symmetric and asymmetric situations. It turns out that in the former case, the optimal tree can be computed in polynomial time, while the latter case is NP-hard. For the latter case, we design an approximation algorithm using a reduction to Connected Facility Location (CFL).

2.1 The Symmetric Case: $SymT$

Consider any instance $\mathcal{I}$ of the problem on $G = (V, E)$, where the terminals are $W = \{v_1, \ldots, v_k\} \subseteq V$, and the upper bound (or *threshold*) for terminal v_i is $b(i) \geq 0$. A solution to this instance of $SymT$ is a tree $T = (V', E')$ with $W \subseteq V' \subseteq V$, where each edge $e \in E(T)$ is allocated a bandwidth x_e. The cost of the tree T is $\sum_{e \in T} c_e x_e$, and is denoted by $C(T)$. The objective is to find a tree such that $C(T)$ is minimum.

Before we begin, let us look at the structure of any solution to this problem. It is clear that any demand between a pair of vertices must be routed on the unique simple path between them in the tree. We claim that this determines the bandwidth allocation x_e on the edges and hence the cost $C(T)$. Indeed, given a tree T and an edge $e \in T$, let L_e and R_e be the trees obtained by deleting

e, and let $b(L_e)$ and $b(R_e)$ be the sums of the thresholds for the terminals in the two trees. Now the edge e must have bandwidth at least $\min\{b(L_e), b(R_e)\}$, since it is possible to set up a valid traffic matrix which requires this amount of flow to pass through e, in which all the terminals on the lighter side want to send flow to the heavier side. Furthermore, this bandwidth is sufficient as well, and hence we can set x_e to be $\min\{b(L_e), b(R_e)\}$. Finally, note that each leaf of T must be a terminal, since no flow will ever pass through a non-terminal leaf, and hence it can be deleted.

We can now construct a directed tree $\vec{T}$ by directing the edges of T towards the lighter side as follows: if $b(L_e) < b(R_e)$, we direct the edge e towards L_e; if $b(R_e) < b(L_e)$ then we direct it towards R_e; and if $b(L_e) = b(R_e)$ then we direct it towards the component which contains some fixed leaf l. Let us now state some useful facts about $\vec{T}$.

Lemma 2.1 *There exists a unique vertex $r \in \vec{T}$ with in-degree 0. Furthermore, every edge in $\vec{T}$ is directed away from r.*

Proof: Any directed tree has at least one vertex r with in-degree 0. Let $e = (x, y)$ such that r is closer to x than to y in T. We want to show that e is directed from x to y in $\vec{T}$. This implies that every other vertex in $\vec{T}$ has in-degree exactly 1, which implies the uniqueness of r as well.

Let P be the path in T from r to x. We know that the first edge $e' = (r, u)$ of P is directed away from r. So, if $L_{e'}, R_{e'}$ are the two components of $T - e'$ and $r \in L_{e'}$, then $b(R_{e'}) \leq b(L_{e'})$. Let L_e, R_e be the two components of $T - e$ such that $x \in L_e$. It is easy to see that $L_{e'} \subseteq L_e$ and hence $R_e \subseteq R_{e'}$. Combining these facts, we get that $b(R_e) \leq b(R_{e'}) \leq b(L_{e'}) \leq b(L_e)$. If $b(R_e) < b(L_e)$, then e is directed from x to y and so we are done. The other possibility is $b(R_e) = b(L_e)$, in which case we must have $R_e = R_{e'}$, $L_e = L_{e'}$ and $b(L_{e'}) = b(R_{e'})$. But since e' is directed towards $R_{e'}$, it must be the case that $l \in R_{e'}$. This, in turn, implies that $l \in R_e$ and thus e must also be directed towards R_e, proving the result. ∎

If the tree T is now imagined as being rooted at the vertex r, the above lemma implies that all edges are directed away from the root. It is easy to see that the bandwidth of an edge connecting a node $u \in T$ to its parent $p_T(u)$ is now exactly $b(T_u)$, where T_u is the subtree rooted at u, and hence $C(T) = \sum_{u \in T} c_{(u, p_T(u))} b(T_u)$. For the rest of this section, let us interpret c_e as being the *length* of an edge. The following lemma provides an alternate way to look at the cost of the tree T, given the vertex r.

Lemma 2.2 *For a tree T with edge-lengths c_e and root r as above, $C(T) = \sum_{u \in T} b(u)\, d_T(r, u)$, where d_T is the distance in T according these edge-lengths. Furthermore, for any other vertex $v \in T$, $C(T) \leq \sum_{u \in T} b(u)\, d_T(v, u)$.*

Proof: The cost of T is

$$\sum_{u \in T} c_{(u,p(u))} \sum_{v \in T_u} b(v) = \sum_{v \in T} b(v) \sum_{u \text{ above } v} c_{(u,p(u))}.$$

However, the inner sum is exactly the distance from v to r with edge-lengths c_e, which proves the first part of the lemma. For the second part, it suffices to show that for every directed edge

$e = (x, y)$ in $\vec{T}$, the weighted sum of the distances from x is no larger than from y. Let L_e, R_e be the two components of $T - e$, with $x \in L_e$. Now

$$\begin{aligned}
&\sum_{u \in T} b(u)\ (d_T(x, u) - d_T(y, u)) \\
&\quad = \sum_{u \in L_e} b(u)\,(d_T(x, u) - d_T(y, u)) \\
&\qquad + \sum_{u \in R_e} b(u)\,(d_T(x, u) - d_T(y, u)) \\
&\quad = -\sum_{u \in L_e} b(u)c_e + \sum_{u \in R_e} b(u)c_e \\
&\qquad = c_e\,(b(R_e) - b(L_e)) \ \leq\ 0,
\end{aligned}$$

where the last inequality follows from the fact that $b(R_e) \leq b(L_e)$, since e is directed towards R_e. ■

Theorem 2.3 *The optimal tree solution for the symmetric case can be computed in $O(S(G))$ time, where $S(G)$ is the time required to compute all-pairs shortest-paths on graph G.*

Proof: For the optimal tree T^*, there exists a vertex $r \in T^*$ such that the cost of the shortest-path tree T connecting r to the set W is exactly the cost of T^*, and thus is just as good a solution as T itself. But now a natural algorithm for finding the optimal tree suggests itself: for each vertex $u \in V$, we find the shortest-path tree T^u connecting u to the vertices of W. We output the tree among these for which the cost $C(T^u)$ is the least. The facts above immediately imply the following theorem. To see that this is optimal, note that $C(T^u) \leq C(T^r)$. This, by the second part of Lemma 2.2, is at most $\sum_u b(u) d_{T^r}(r, u) \leq \sum_u b(u) d_{T^*}(r, u) = C(T^*)$. ■

2.2 The Asymmetric Case: $AsymT$

While we could efficiently compute the optimal solution in the symmetric case, the situation in the asymmetric case is more complicated, since the problem $AsymT$ turns out to be NP-hard.

Theorem 2.4 *The problem $AsymT$ is strongly NP-hard and max-SNP hard.*

Proof: We give an approximation-preserving reduction from the minimum Steiner tree problem, which is strongly NP-hard [9]. Given a Steiner tree instance $\mathcal{I}$ on the graph $G = (V, E)$ with edge weights c_e and required vertices R, we create an instance $\mathcal{I}'$ of $AsymT$, where the edge weights are c_e, R is the set of terminals, each with thresholds $(b_{in} = \infty, b_{out} = 1)$.

Consider any tree solution T for $\mathcal{I}'$, and an edge $e \in T$. Deleting e creates two subtrees L_e and R_e. Clearly, the bandwidth allocated to the edge e must be $b_{out}(L_e) + b_{out}(R_e)$, since it is possible to set up a valid traffic matrix where each terminal in L_e sends flow to some terminal in R_e, and *vice versa*. Furthermore, this is the maximum possible flow across the edge e. Since this quantity is simply $b_{out}(W) = |R|$, the cost of the tree is $|R| \sum_{e \in T} c_e$. Finding a tree minimizing this cost is the same as finding the optimal Steiner tree, which completes the proof. ■

However, we can reduce finding the optimal tree to the following NP-hard facility location problem.

Definition 2.5 *An instance of* Connected Facility Location *is specified by a graph $G = (V, E)$ with edge lengths c_e, a set of* demand nodes D *where each vertex $j \in D$ has a demand $d_j > 0$,* facility costs f_i *associated with opening a facility at location $i \in V$, and a parameter $M \geq 1$. The objective is to open a set of facilities F, to assign some open facility $i(j)$ to each demand node j, and to connect up all the vertices in F with a Steiner tree T so as to minimize*

$$\sum_{i \in F} f_i + \sum_{j \in V} d_j c_{i(j)j} + M \sum_{e \in T} c_e,$$

where c_{uv} denotes the length of the shortest path between u and v in V.

The reduction involves some amount of detail, which we defer to the Appendix. However, let us briefly sketch some of the underlying ideas. Given a tree solution T, we can infer the allocation on an edge e thus: if deleting the edge e creates subtrees L_e and R_e, then $x_e = \min\{b_{in}(L_e), b_{out}(R_e)\} + \min\{b_{in}(R_e), b_{out}(L_e)\}$. We can again define a directed tree $\vec{T}$ by replacing each undirected edge by *two* directed edges, where the directions indicate which of the subtrees attain the minima in the above expression. It can be shown that the edges of T where both directed edges have opposite directions forms a connected component of T which corresponds to the Steiner tree in CFL, while those where the edges have the same direction behave in a manner similar to the $SymT$ case and correspond to the connection costs.

2.2.1 Approximation algorithms for CFL

Consider an instance $\mathcal{I}$ of the Connected Facility Location problem, as defined in Definition 2.5. Let us assume that we know some vertex v which is opened as a facility in the optimal solution. (We can discharge this assumption by trying out all possible vertices v.) It is not difficult to check that we can formulate the problem as the following 0-1 integer program, where y_i indicates if a facility is opened at location i, x_{ij} indicates whether demand point j is assigned to facility i, and z_e indicates whether edge e is part of the Steiner tree connecting the opened facilities.

$$\min \quad \sum_{i \in V} f_i y_i + \sum_{j \in D} d_j \sum_{i \in V} x_{ij} c_{ij} + M \sum_{e \in E} c_e z_e \tag{IP1}$$

$$\text{s.t.} \quad \sum_{i \in V} x_{ij} = 1 \qquad \text{for all } j \in D \tag{2.1}$$

$$x_{ij} \leq y_i \qquad \text{for all } j \in D, i \in V \tag{2.2}$$

$$\sum_{i \in S} x_{ij} \leq \sum_{e \in \delta(S)} z_e \quad \text{for all } S \subseteq V, v \notin S \text{ and } j \in D \tag{2.3}$$

$$x_j, y_i, z_e \in \{0, 1\}$$

As usual, we relax (IP1) to a linear program (LP1) by replacing the 0-1 requirements by the constraints $x_{ij}, y_i, z_e \geq 0$. We can show that (LP1) can be solved in polynomial time using the ellipsoid algorithm.

Theorem 2.6 *The optimal (possibly fractional) solution to (LP1) can be rounded to an integer solution increasing the cost by at most a factor of 12. Furthermore, in the case where all the* $f_i = 0$*, we can get an improved approximation guarantee of 9.002.*

Proof: The basic idea of the rounding algorithm is similar to the one given by Shmoys et al. [22]. The algorithm first performs a filtering step [16, 17] followed by clustering, after which a facility is chosen from each cluster. By the properties of the filtering, the cost of opening this facility and connecting the demand points to it is not much more than the fractional cost. However, showing that a near-optimal Steiner tree connecting these facilities has low cost requires more work. To show this, an auxiliary graph is created in which some of the potential facilities in each cluster are contracted, and an approximate Steiner tree is constructed in this collapsed graph. Finally, this tree is extended to a Steiner tree of low cost in the original graph, completing the proof. The details are given below.

Filtering: Let x, y, z be the optimal fractional solution to this LP. Let $0 < \alpha < 1$ be a constant whose value we will fix later. For each demand point j, define $c_j(\alpha)$ as follows: let π be a permutation such that $c_{\pi(1)j} \leq c_{\pi(2)j} \leq \cdots \leq c_{\pi(n)j}$. Define $i^* = \min\{i' : \sum_{i=1}^{i'} x_{\pi(i)j} \geq \alpha\}$, and $c_j(\alpha)$ to be $c_{\pi(i^*)j}$. We will use the following fact extensively:

$$(1-\alpha)c_j(\alpha) \;\leq\; \sum_{i \geq i^*} x_{\pi(i)j} c_{\pi(i)j} \;\leq\; \sum_{i=1}^{n} c_{ij} x_{ij}. \tag{2.4}$$

Let us define a new solution x', y', z' as follows : for each demand point j, set $\alpha_j = \sum_{i: c_{ij} \leq c_j(\alpha)} x_{ij}$. Define x'_{ij} to be x_{ij}/α_j if $c_{ij} \leq c_j(\alpha)$, and 0 otherwise. For each i, define $y'_i = \min\{1, y_i/\alpha\}$. For each edge e, define $z'_e = z_e/\alpha$. It is easy to verify that this new solution is feasible for (LP1). Moreover, (2.4) tells us that we can assign a demand j to any facility i such that $x'_{ij} > 0$ without incurring a loss of more than $1/(1-\alpha)$ in the assignment cost.

Clustering: We now define a family of disjoint *clusters*, where each cluster is a set of demand and facility points. The cluster C_1 is defined by choosing a demand point j called the *center* of the cluster C_1, which is the point with the minimum $c_j(\alpha)$. To do this, let $\beta > 1$ be a constant whose values will be decided later. Let F_1 be the set of facilities i such that $x'_{ij} > 0$. All the facilities in F_1 are added to the cluster C_1, as are all demand points j' such that $x'_{ij'} > 0$ for some $i \in F_1$. (In particular, $j \in C_1$). Furthermore, if there is a demand point j'' such that one of the facilities in F_1 is within a distance $\beta c_{j''}(\alpha)$ of j'', then j'' is *reassigned* to the closest facility in F_1, and is also added to C_1. The vertices in C_1 are now deleted, the clusters are formed in the remaining graph until it is depleted, at which time we have disjoint clusters $C_1, C_2, \ldots, C_r$.

The fact to notice here is that if there are two clusters C_p and C_q with centers j and j', then any facility in F_p is at least at distance $(\beta - 1)\max\{c_j(\alpha), c_{j'}(\alpha)\}$ from any facility in F_q.

Picking the facilities: For each cluster C_k, we pick the facility $i^*_k \in F_k$ with the lowest cost f^*_k, and assign all the demand points in C_k to this facility. By (2.1), (2.2) and a simple averaging argument, the cost of opening this facility will be at most $\sum_{i \in F_k} f_i y'_i$, and hence the total facility cost will be at most $\sum_{i \in V} f_i y_i/\alpha$.

We also claim that this does not increase the assignment cost substantially. Indeed, if j' is a demand point such that $x'_{ij'} > 0$ for $i \in F_k$, the triangle inequality implies that the distance of j'

from any facility in F_k is at most $c_{j'}(\alpha)+2c_j(\alpha) \le 3c_{j'}(\alpha)$. If a demand point j' was reassigned to a facility in F_k, then the j' is at most $\beta c_{j'}(\alpha)+2c_j(\alpha) \le (\beta+2)c_{j'}(\alpha)$ distance from any facility in F_1. Now (2.4) implies that the assignment cost of any demand point will increase by a factor of at most $(\beta+2)/(1-\alpha)$.

The Steiner tree: We will now construct a Steiner tree connecting the opened facilities whose cost is not much more than $\sum_e c_e z_e$. For each cluster C_k, the nodes in the set F_k are contracted into a *supernode* ϕ_k, and let the resulting contracted graph be denoted by G_c. Since the sets F_i are disjoint, each supernode will be distinct, and let S_F be the set of these r supernodes. If the vertex v is not included in any of the supernodes, it is also added to S_F. The variables x' can be defined on this new graph in the natural fashion by setting $x'_{\phi_k j} = \sum_{i \in F_k} x_{ij}$. Let S be any subset of nodes in G' such that $S_F \cap S$ and $S_F - S$ are both non-empty, and assume that $v \notin S$. (This is without loss of generality, since we can always replace S by $V - S$.) Now (2.1) and (2.3) imply that $\sum_{e \in \delta(S)} z_e \ge \sum_{i \in S} x'_{\phi_k j} = 1$, where ϕ_k is a supernode in S and j is the center of the cluster C_k. Thus z'_e is a feasible fractional solution to the LP relaxation of the Steiner tree problem on G' with S_F as the set of required nodes. Since the integrality gap of the Steiner tree LP is at most 2, we can use any algorithm to find a good approximation to the minimum Steiner tree T' connecting vertices in S_F with cost at most $2\sum_e c_e z'_e$.

The final step is to use T' to find a cheap Steiner tree T in the original graph G connecting the open facilities. This becomes slightly involved due to the following fact: if we interpret the tree T' in G, it may be the case that the edges do not form a connected subgraph. This is because edges incident to ϕ_k in G' may be actually be incident to different vertices of F_k in G. To handle this problem, we add edges to connect the center of the cluster C_k to all these vertices and to i_k^*, the facility opened in cluster C_k. These edges, along with the edges of T' form the Steiner tree T.

It now remains to show that the cost of T is not much more than that of T', which we will show by charging the cost of these edges to the edges of T'. Let us direct the edges in T' to form an outgoing arborescence from v (or the supernode containing it). If we consider F_k (with center j), there is exactly one facility i_k which has an incoming arc in T', while there are outgoing arcs from facilities $i_{k_1}, \ldots, i_{k_r}$.

First, we will charge the edges connecting j to each of the i_{k_l}, which cost at most $c_j(\alpha)$, to the path in T' connecting ϕ_k to the next supernode, which must be of length at least $(\beta-1)c_j(\alpha)$. It is easy to see that these paths in T' must be disjoint, and hence the total cost of these edges is at most the cost of T'. The cost of the path $\langle j, i_k \rangle$ can be charged to the incoming edge, and the path $\langle j, i_k^* \rangle$ can be charged to either an incoming edge or an outgoing one (in the case of the root). Summing up, the cost of T is at most $1+4/(\beta-1)$ times the cost of T', and hence at most $2(\beta+3)/\alpha(\beta-1)$ times $\sum_e c_e z_e$.

This gives us a $\max\left\{\frac{1}{\alpha}, \frac{\beta+2}{1-\alpha}, \frac{2(\beta+3)}{\alpha(\beta-1)}\right\}$ factor approximation algorithm for the problem. Setting α to be 0.45 and β to 3.86, we get a constant about 10.66. In the useful case when there are no facility costs, we can open the facility i_k in cluster C_i instead of i_k^*. So, the approximation ratio becomes $\max\left\{\frac{\beta+2}{1-\alpha}, \frac{2(\beta+1)}{\alpha(\beta-1)}\right\}$. Setting $\alpha = 0.44, \beta = 3.04$, we get a constant slightly less than 9.002. ■

This also gives the 9.002-approximation algorithm for $AsymT$. (We have not tried to optimize the constants in the above analysis; better bounds are possible using randomization.) For the source-limited case where each threshold is $(b_{in} = \infty, b_{out} = \cdot)$, the arguments of Theorem 2.4

show that the optimal tree solution corresponds exactly to the optimal Steiner tree connecting the terminal set, and hence can be approximated to within 1.55 [19].

3 Unsplittable and Fractional Graph Solutions

3.1 The Symmetric Case

In this section, we show that a solution to $SymT$ is a constant factor approximation to $SymG$. In fact, we show something stronger; i.e., a solution to $SymT$ is within $2(1-1/k)$ of the optimal fractional solution to $SymF$.

To prove the result, let us define the *Pairwise Demands* problem $PD(\lambda)$. An instance of this problem is also given by $G=(V,E)$ with edge costs c_e, and set W of k terminals, but now the objective is to find the least cost allocation of bandwidth x_e to the edges in G so that λ units of flow can be sent between *every* pair of terminals. This can be solved in polynomial time by taking the union of the shortest paths between each pair of terminals, where x_e is λ times the number of the paths using e.

In the rest of this section, let us assume that $b(v)=1$ for all $v \in W$ in an instance $\mathcal{I}$ of $SymT$ under consideration. Since we are finding an optimal *tree* solution, it can be shown that given an instance $\mathcal{I}'$ of $SymT$ with arbitrary thresholds, we can create an instance $\mathcal{I}$ where all thresholds are 1 by replacing a terminal of threshold $b(i)$ by a star of $b(i)$ nodes (with zero-cost edges), and the optimal tree solution of $\mathcal{I}$ and $\mathcal{I}'$ remain the same. (This assumption will merely be required for the proofs, and not for the algorithms.)

We now argue that the optimal solution to $PD(1/(k-1))$ is close to the optimal solutions for both $SymF$ and $SymT$ (where $b(i)=1$ for all $i \in W$), and since

$$OPT(SymF) \leq OPT(SymG) \leq OPT(SymT),$$

we will have shown that $SymT$ is close to $SymG$.

Theorem 3.1 *For any graph $G=(V,E)$ with edge-costs c_e, and set W of k terminals, it holds that:*

$$\begin{aligned} OPT(SymF) \leq OPT(SymG) \leq\ & OPT(SymT) \\ \leq\ & 2(1-\tfrac{1}{k})\, OPT(PD(\tfrac{1}{k-1})) \\ \leq\ & 2(1-\tfrac{1}{k})\, OPT(SymF). \end{aligned}$$

Proof: The inequalities on the first line are obvious. For the last inequality, consider an optimal solution $\mathcal{S}$ to $SymF$. Since each terminal has unit threshold, the demand vector $d_{ij}=1/(k-1)$ for $i,j \in W$ is a a valid traffic matrix, and hence $\mathcal{S}$ is a feasible solution to $PD(\frac{1}{k-1})$ as well, and hence $OPT(PD(\frac{1}{k-1})) \leq OPT(SymF)$.

Now consider an optimal solution $\mathcal{S}$ to $PD(\frac{1}{k-1})$. For a terminal $i \in S$, let F_i denote the portion of the flow $\mathcal{S}$ corresponding to sending $1/(k-1)$ units of flow from i to all other terminals. Also, for each terminal i, let T_i be the shortest-path tree from i in the graph G (with edge lengths c_e), and $cost(T_i) = \sum_{u \in W} d_G(i,u) = \sum_{u \in W} d_{T_i}(i,u)$ denote the sum of the cost of the paths in T_i from i to all other terminals in W. Now, note that the optimal way of sending $1/(k-1)$ units of flow from i to all terminals in S is to send them along the unique path from i to each terminal in the tree T_i. This implies that $(k-1)cost(F_i) \geq cost(T_i)$. But since the flow between i and j is being counted in both F_i and F_j, $cost(F_1) + \cdots + cost(F_k) = 2cost(\mathcal{S})$, and thus $cost(T_1) + \cdots + cost(T_k) \leq 2(k-1)cost(\mathcal{S})$. Now if $cost(T_1) \leq cost(T_i)$ for all $i \in W$, then $cost(T_1) \leq \frac{2(k-1)}{k}cost(\mathcal{S})$.

Finally, we must show that there is a solution to $SymT$ with cost at most $cost(T_1)$. But this is simple to see by an argument similar to one used earlier: the number of times an edge e is counted in T_i is just the number of nodes of W in one of the subtrees L_e created by deleting e, and hence at least $\min\{b(L_e), b(R_e)\}$, which is the required bandwidth allocation to e in T. This shows that T_1 itself is a solution to $SymT$ with cost at most $cost(T_1)$, proving the theorem. ∎

It can be seen that Theorem 3.1 implies that the optimal solution to $PD(1/(k-1))$ is also within a factor at most 2 of $OPT(SymF)$, which gives us a 2-approximation algorithm to $SymF$.

3.1.1 When each vertex routes on a tree

The routing strategies used in most current systems are tree-routing protocols, where each terminal s has a tree T_s (usually the shortest-path tree rooted at s) connecting s to all the terminals. When s wants to send transmissions to a subset S of the terminals, it sends them unsplittably along the edges of this tree. As discussed in the introduction, tree-routing has advantages like simplicity and scalability. To cast this notion in our framework, we can search for the following restricted type of solution to an instance of $SymG$: instead of specifying the paths P_{ij} explicitly, we specify them by giving a tree T_i for each terminal i. The transmissions between i and other vertices take place along the edges of T_i, and hence the collection of trees implicitly defines the paths P_{ij}.

This additional condition is clearly vacuous if we are seeking an optimal solution to $SymT$, since the union of the paths P_{ij} is clearly a tree. Interestingly, it turns out that if we are solving $SymG$ and want each terminal i to route unsplittably on a tree T_i, then there is an optimal solution in which all T_i are the same — we cannot do better than simply to solve the corresponding instance of $SymT$ on the terminals.

Theorem 3.2 *If $\mathcal{S}$ is an optimal solution to an instance $\mathcal{I}$ of $SymG$ under the constraint that every terminal routes along a tree, and $\mathcal{S}'$ is the optimal solution to $SymT$ on $\mathcal{I}$, then $C(\mathcal{S}) = C(\mathcal{S}')$.*

Proof: Let $\mathcal{S}$ be the optimal solution to $SymG$ under the restriction that every terminal i routes its flow along a fixed tree T_i. This implicitly defines a (simple) path P_{ij} between i and j (which is the same as P_{ji}, since we are in the symmetric case). Let $\mathcal{P}_i$ be the set of simple paths defined by T_i, and $n_i(e)$ is the number of paths in P_i using edge e. Also, let x_e be the bandwidth allocated to edge e by $\mathcal{S}$. We want to show that $\sum_e c_e x_e \geq C(\mathcal{S}')$. As in the previous section, we can assume that $b(i) = 1$ for all terminals i in the proofs.

To lower bound $\sum c_e x_e$, we make an auxiliary graph $G_e = (W, E_e)$ for each e, where E_e contains the edge (i,j) if and only if P_{ij} contains e. Note that the degree of i in G_e is $n_i(e)$. It is clear that x_e is the maximum fractional matching on G_e. We want to construct a feasible fractional matching by assigning weights w_f to each edge $f \in E_e$, such that if E_e^i is the set of edges in G_e incident to i, then $\sum_{f \in E_e^i} w_f \leq 1$. We will then show that the cost of solution corresponding to this matching is large, and hence the cost corresponding to x_e must be larger still. For any terminal i and edge e, let us define $y_i(e) = \min\{n_i(e), k - n_i(e)\}$, and for a subgraph T_i, define $y(T_i) = \frac{1}{k}\sum_{e \in T_i} c_e y_i(e)$.

Claim 3.3 $\sum_i y(T_i) \leq \sum_e c_e x_e = C(\mathcal{S})$.

Proof: To prove the claim, it suffices to show that $\frac{1}{k}\sum_i y_i(e) = \frac{1}{k}\sum_i \min\{n_i(e), k - n_i(e)\} \leq x_e$ for all edges e; and hence it suffices to construct a fractional matching in G_e whose value is at least the left side of this inequality. To this end, let us define

$$w_f = \frac{1}{k}\left(\frac{y_i(e)}{n_i(e)} + \frac{y_j(e)}{n_j(e)}\right)$$

for each edge $f = (i, j)$ in G_e. It is easy to verify that $\sum_{f \in E_e} w_f = \frac{1}{k}\sum_i y_i(e)$. We also have to ensure that w_f satisfies $\sum_{f \in E_e(i)} w_f \leq 1$ and is a fractional matching in G_e. Let $N(i)$ denote the neighbors of i in G_e.

$$\begin{aligned}
\sum_{f \in F_i} w_f = \frac{1}{k} \sum_{j \in N(i)} & \left(\frac{\min\{n_i(e), k - n_i(e)\}}{n_i(e)} \right. \\
& \qquad \left. + \frac{\min\{n_j(e), k - n_j(e)\}}{n_j(e)} \right) \\
\leq \frac{1}{k} \sum_{j \in N(i)} & \left(\frac{k - n_i(e)}{n_i(e)} + \frac{n_j(e)}{n_j(e)} \right) \\
= \frac{1}{k} (k - n_i(e) & + n_i(e)) \;=\; 1.
\end{aligned}$$

■

To complete the proof of Theorem 3.2, it suffices to show that $y(T_i) \geq C(\mathcal{S}')/k$, which along with Claim 3.3 would imply that $C(\mathcal{S}') \leq C(\mathcal{S})$. Note that T_i is a tree joining all the terminals, and by the arguments of Section 2.1, a feasible bandwidth allocation on the edge e is the smaller of the numbers of terminals in L_e and R_e. This, in turn, can be seen to be bounded above by $\min\{n_i(e), k - n_i(e)\}$, and hence the optimal tree solution has cost $C(\mathcal{S})$ at most $\sum_{e \in T_i} c_e \min\{n_i(e), k - n_i(e)\} \leq k\, y(T_i)$, which completes the proof. ■

3.2 The Asymmetric Case

In the asymmetric case, we have results for the setting in which all thresholds are of the form $(b_{in} = \infty, b_{out} = \cdot)$. In this case, we show that the problems $AsymT$ and $AsymG$ are the same, and are in turn within a factor of 2 of $AsymF$. Let $B = \sum_{u \in W} b_{out}(u)$ in this section.

Theorem 3.4 *For any instance $\mathcal{I}$ of $AsymG$, with all thresholds being $(b_{in} = \infty, b_{out} = \cdot)$, there is an optimal tree solution.*

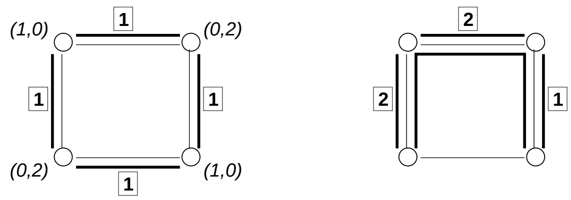

Figure 2: *Example showing that $AsymG$ (on the left) may not equal $AsymT$ (right), where numbers in italics denote (b_{in}, b_{out}) values, numbers in boxes are bandwidth allocations, and thick black lines are assigned paths.*

Proof: Consider an instance $\mathcal{I}$ of $AsymG$ with all thresholds being $(b_{in} = \infty, b_{out} = \cdot)$. Let T^* be the optimal Steiner tree connecting W, and let $C^* = \sum_{e \in T^*} c_e$ be its cost. We will show that, for any solution $\mathcal{S}$ to $\mathcal{I}$ assigning bandwidth x_e to edge e, its cost $C(\mathcal{S}) = \sum_{e \in E} c_e x_e \geq B\, C^*$.

Let $\mathcal{P}$ be the set of $k(k-1)$ paths in $\mathcal{S}$, one path between each distinct (ordered) pair of nodes in W. Let $\mathcal{P}_e$ denote the set of paths in $\mathcal{P}$ which contain e. We claim that x_e is equal to the sum of $b_{out}(i)$ for $\{i \mid \exists j\, s.t.\, P_{ij} \in \mathcal{P}_e\}$; i.e., the sum of $b_{out}(i)$ for those i which are the left ends of some (directed) path in P_e. Indeed, note that we can set up flows from each of these vertices so that exactly the claimed amount of flow passes through x_e, but no more.

If $W = \{1, \ldots, k\}$, consider the subgraphs $T_1, \ldots, T_k$ where T_i is the union of the paths P_{ij} which connect i to the other vertices of W. Define $c(T_i) = \sum_{e \in T_i} c_e$. Since the tree T_i connects all the nodes in W, $c(T_i) \geq C^*$. Hence $\sum_{i=1}^{k} b_{out}(i)\, c(T_i) \geq B\, C^*$. Finally, we show that $\sum b_{out}(i)\, c(T_i) = \sum_{e \in E} c_e x_e$. Look at an edge e, and the paths $\mathcal{P}_e$ that pass through it. Since we have already shown that x_e is the sum of the $b_{out}(i)$ of the set of left endpoints i of these paths, it suffices to show that e appears in the P_i corresponding to these very i's. But this is immediate by the definition of T_i. This shows that the cost of the optimal solution is at least $B\, C^*$, completing the proof. ∎

Furthermore, we can also show that the optimal unsplittable graph solution is within a factor of 2 of the fractional solution.

Theorem 3.5 *For any instance with all thresholds being $(b_{in} = \infty, b_{out} = \cdot)$,*

$$\begin{aligned} OPT(AsymF) &\leq OPT(AsymT) \\ &= OPT(AsymG) \leq 2OPT(AsymF). \end{aligned}$$

Proof: The first inequality is trivial, and the equality follows from the previous theorem. To see the final equality, note that $1/B$ times the optimal solution $\mathcal{S}$ to $AsymF$ is also feasible for the LP relaxation of the Steiner tree problem and hence $\mathcal{S}$ has cost at least $B/2\ c(T^*)$, where T^* is the optimal Steiner tree. But by the arguments of Theorem 2.4, $OPT(AsymT) = B\ c(T^*)$, and the theorem follows. ∎

The result of Theorem 3.4 does not hold for the general case when arbitrary values of b_{in} and b_{out} are allowed, as Figure 2 shows. We also do not know of an analogue of Theorem 3.5 for the general case, and offer this as an open problem.

3.3 Fractional Solutions

Finding optimal solutions to the fractional variants appears to be quite difficult; and this is perhaps surprising given the relationship to standard multicommodity flow problems. Part of the complication arises from the fact that the demands are not fixed, as in multicommodity flow, but allowed to range over all valid traffic matrices.

Specifically, we prove the following.

Theorem 3.6 *The problem $AsymF$ is co-NP hard to solve on directed graphs.*

The question of whether finding optimal fractional solutions is hard for undirected graphs (either in the symmetric or the asymmetric case) remains as yet unresolved.

4 Network Design with capacities

In the previous sections, we assumed that any amount of bandwidth could be allocated to any edge in G. A more general situation is one in which each edge has a capacity u_e, and we must satisfy $x_e \leq u_e$. We can show that the analogues of $SymT$ and $SymG$ in the capacitated case are NP-hard.

Theorem 4.1 *The problem of deciding whether a given capacitated version of $SymT$ or $SymG$ has a feasible solution is NP-hard, even when all the capacities are at most 2.*

This result, along with the result of Section 3.3, suggests that checking feasibility of even the fractional version of this problem may be hard. The following theorem gives a simple approximation algorithm for the fractional case of $SymG$ which gives a solution which is within a constant factor of the optimum, and while violating the capacity of any edge by a constant factor.

Theorem 4.2 *There is a polynomial time algorithm for the the capacitated version of $SymG$ which outputs a solution whose cost is within a constant factor of the optimal solution cost. Further, this solution violates any edge capacity by at most a constant.*

Proof: Before stating the approximation algorithm, let us observe a simple fact about matchings in graphs. Given vectors $M_1, \ldots, M_r$ of the same dimension, we say that a vector M is an *approximate convex combination* of these vectors if there exist real non-negative constants $\lambda_1, \ldots, \lambda_r$ such that $M = \sum_{i=1}^{r} \lambda_i M_i$ and $\sum_{i=1}^{r} \lambda_i = 3/2$. We say that M_1 *dominates* M_2 if each coordinate of M_1 is at least the corresponding coordinate of M_2.

Given an undirected graph $G' = (V', E')$, the fractional matching problem is to assign non-negative edge weights y_e to each edge in E' such that for every vertex $v \in V'$, $\sum_{e \in \Gamma(v)} y_e \leq 1$, where $\Gamma(v)$ is the set of edges incident with v. Note that a matching in G' is a fractional matching where each y_e is 0 or 1. We can view each (fractional or integral) matching as a vector of length $|E'|$.

Lemma 4.3 *Each fractional matching in the graph G' (viewed as a vector of length $|E'|$) is dominated by an approximate convex combination of matchings in G'.*

Proof of Lemma 4.3: Consider the fractional matching polytope $\mathcal{P}$ of G'. It is well known that if M is a vertex of $\mathcal{P}$, then each coordinate of M is $0, 1/2$ or 1 [18]. Clearly, it is enough to prove the lemma for fractional matchings which are vertices of $\mathcal{P}$. Let M be a vertex of $\mathcal{P}$. Note that if E'' are the edges in M with non-negative weight, then E'' is a set of vertex disjoint edges (call this E_1'') and odd cycles (call this E_2''). Each edge in E_1'' gets weight 1 and edge in E_2'' gets weight 1/2. It is easy to see that E_1'' is a matching and E_2'' can be expressed as union of three matchings M_1, M_2, M_3. Clearly, $E_1'' \cup M_i$ is also a matching. Now, M is dominated by $1/2 \sum_{i=1}^{3} (E_1'' \cup M_i)$, which proves the lemma. ■

Without loss of generality, let us assume that the threshold of each terminal is 1. (Otherwise, replace a terminal with threshold $b(i)$ with a "star" having $b(i)$ leaves, each leaf having threshold 1). We consider the following pairwise demand between the terminal nodes W: for each $v_i, v_j \in W$, let demand between them be $1/k$ (recall that $k = |K|$). Let f denote an optimal flow for this pairwise demand. Let $f_{i,j}$ be the flow between v_i and v_j and $f(e)$ be the flow on an edge e.

We will allocate a bandwidth $x_e = 3f(e)$ on each edge. Clearly, this solution satisfies the requirements of the theorem, and it suffices to show that this is a feasible solution. For every pair of vertices $i, j \in W$, we define a flow $f'_{i,j}$ between them as follows : first send $1/k$ units of flow from i to each terminal r along $f_{i,r}$, and then from each terminal r, send $1/k$ units of flow to j using f_{rj}.

Let G' be the complete graph on W as vertices. Note that any set of pairwise demands between the nodes in W is a fractional matching in G'. We first show how to route any integral matching in G' when the reservation on an edge e is just $2f(e)$. Let M be a matching in G': for every edge (i, j) in this matching, send one unit of flow between them along $f'_{i,j}$. It is easy to see that this needs at most $2f(e)$ units of bandwidth on an edge e, simply because the maximum traffic that gets routed via any vertex is at most 2. Since Lemma 4.3 shows that any fractional matching is an approximate convex combination of integral matchings, this implies that any pairwise demand can be routed if the reservation on an edge is $3f(e) = x_e$, which proves the theorem. ■

Acknowledgments

We thank Éva Tardos for several useful discussions.

References

[1] Karen I. Aardal, Fabian Chudak, and David B. Shmoys. A 3-approximation algorithm for the k-level uncapacitated facility location problem. *Information Processing Letters*, 72:161–167, 1999.

[2] Matthew Andrews and Lisa Zhang. The access network design problem. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 40–49, 1998.

[3] Esther M. Arkin, Magnús M. Halldórsson, and Refael Hassin. Approximating the tree and tour covers of a graph. *Information Processing Letters*, 47(6):275–282, 1993.

[4] Baruch Awerbuch and Yossi Azar. Buy-at-bulk network design. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 542–547, 1997.

[5] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 378–388, 1999.

[6] Bruce Davie and Yakov Rekhter. *MPLS: Technology and Applications*. Morgan Kaufmann Publishers, 2000.

[7] Nicholas G. Duffield, Pawan Goyal, Albert G. Greenberg, Partho P. Mishra, K.K. Ramakrishnan, and Jacobus E. van der Merwe. A flexible model for resource management in virtual private networks. In *Proceedings of the ACM SIGCOMM, Computer Communication Review*, volume 29, pages 95–108, 1999.

[8] Shivi Fotedar, Mario Gerla, Paola Crocetti, and Luigi Fratta. ATM virtual private networks. *Communications of the ACM*, 38(2):101–109, 1995.

[9] Michael R. Garey and David S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. W. H. Freeman and Company, San Francisco, 1979.

[10] Michel X. Goemans, Andrew V. Goldberg, Serge Plotkin, David B. Shmoys, Éva Tardos, and David P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 1994.

[11] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*. Springer-Verlag, Berlin, 1988.

[12] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Hierarchical placement and network design problems. In *Proceedings of the 41th Annual IEEE Symposium on Foundations of Computer Science*, pages 603–612, 2000.

[13] Kamal Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 448–457, 1998.

[14] Kamal Jain and Vijay Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1999.

[15] David R. Karger and Maria Minkoff. Building Steiner trees with incomplete global knowledge. In *Proceedings of the 41th Annual IEEE Symposium on Foundations of Computer Science*, pages 613–623, 2000.

[16] Jyh-Han Lin and Jeffrey Scott Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44(5):245–249, 1992.

[17] Jyh-Han Lin and Jeffrey Scott Vitter. ϵ-approximations with minimum packing constraint violation (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 771–782, 1992.

[18] László Lovász and Michael D. Plummer. *Matching theory*. North-Holland Publishing Co., Amsterdam, 1986. Annals of Discrete Mathematics, 29.

[19] Gabriel Robins and Alexander Zelikovsky. Improved Steiner tree approximation in graphs. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–779, 2000.

[20] F. Sibel Salman, Joseph Cheriyan, R. Ravi, and Sairam Subramanian. Buy-at-bulk network design: Approximating the single-sink edge installation problem. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 619–628, 1997.

[21] Charlie Scott, Paul Wolfe, Mike Erwin, and Andy Oram. *Virtual Private Networks*. O'Reilly, 1998.

[22] David B. Shmoys, Éva Tardos, and Karen I. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[23] David P. Williamson. *Approximation Algorithms for a Class of Graph Problems*. PhD thesis, MIT, September 1993.

[24] David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A primal-dual approximation algorithm for generalized Steiner network problems. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 708–717, 1993.

A The reduction from $AsymT$ to CFL

In this Appendix, we describe a procedure that transforms instances of $AsymT$ into instances of the Connected Facility Location problem.

Consider any tree solution T to $AsymT$, and look at any edge $e \in T$. As before, let L_e and R_e be the two trees created by deleting e, and let the quantities $b_{in}(T'), b_{out}(T')$ for any subtree T' be defined in the usual way. Let $B = \min\{b_{in}(W), b_{out}(W)\}$. It is not difficult to see that the bandwidth allocation on the edge e is

$$\begin{aligned} x_e &= \min\{b_{in}(L_e), b_{out}(R_e)\} + \min\{b_{in}(R_e), b_{out}(L_e)\} && \text{(A.5)} \\ &= \min\{b_{in}(L_e) + b_{in}(R_e), b_{in}(L_e) + b_{out}(L_e), \\ &\qquad b_{out}(R_e) + b_{in}(R_e), b_{out}(R_e) + b_{out}(L_e)\} && \text{(A.6)} \end{aligned}$$

For the rest of the argument, let us assume that no leaf node in T has both b_{in} and b_{out} equal to 0.

Let us define a directed tree $\vec{T}$ from T, where each undirected edge e is replaced by two directed edges e' and e'' with the same endpoints as e. The idea behind these edges is similar to that for the symmetric case: e' is directed towards L_e if $b_{in}(L_e) > b_{out}(R_e)$; towards R_e if $b_{in}(L_e) < b_{out}(R_e)$; and towards the subtree containing some fixed leaf l if they are equal. The edge e'' is directed similarly by comparing $b_{in}(R_e)$ and $b_{out}(L_e)$. The edge $e \in T$ is called *decisive* if e' and e'' have the same direction in $\vec{T}$, and *indecisive* otherwise. Let us record some observations about T and $\vec{T}$:

Claim A.1 *If e is indecisive, then $x_e = B$.*

Proof: Since the edge e is indecisive, it must be the case that x_e must be the smaller of $b_{out}(L_e) + b_{out}(R_e)$ and $b_{in}(L_e) + b_{in}(R_e)$, which is B. ∎

Now that we know that all indecisive edges have the same bandwidth allocated to them, let us turn our attention to the decisive edges.

Lemma A.2 *Let e be a decisive edge, such that both e' and e'' are directed towards L_e in $\vec{T}$. Then every edge $f \in L_e$ is decisive and directed towards the component L_f of $T - f$ which is contained in L_e.*

Proof: We know that $b_{in}(L_e) \leq b_{out}(R_e)$ and $b_{out}(L_e) \leq b_{in}(R_e)$. Let f be an edge in L_e, and let L_f be the component of $T - f$ contained in L_e. Let L'_f denote $L_e - L_f$ (i.e., the subtree induced by these nodes). Note that $b_{in}(L_f) = b_{in}(L_e) - b_{in}(L'_f)$ and $b_{in}(R_f) = b_{in}(R_e) + b_{in}(L'_f)$, which implies that

$$\begin{aligned} & b_{in}(L_f) - b_{out}(R_f) \\ & \quad = b_{in}(L_e) - b_{out}(R_e) - b_{in}(L'_f) - b_{out}(L'_f) \leq 0. \end{aligned}$$

A similar calculation shows that $b_{out}(L_f) \leq b_{in}(R_f)$. Now, if both the inequalities were strict, we would be done. So assume that $b_{in}(L_f) = b_{out}(R_f)$: this implies $b_{in}(L_e) = b_{out}(R_e)$ and $b_{in}(L'_f) = b_{out}(L'_f) = 0$. Since we have assumed that that no leaf in T has $b_{in} = b_{out} = 0$, the path joining e and f cannot have any degree-three vertex. Also, $b_{in}(L_e) = b_{out}(R_e)$ implies that the special leaf l is in L_e, and by the previous argument, it must be in L_f, and f' will be directed towards L_f. A similar argument shows this for f'' also, completing the proof. ∎

Corollary A.3 *The set of indecisive edges forms a connected subtree of T.*

Proof: If f and g are two indecisive edges such that the unique path in T joining them contains a decisive edge e. Lemma A.2 implies that one of f and g must be decisive, a contradiction. ∎

An example of the structure of the tree T is given in Figure 3, where the thin directed edges are decisive (where the direction indicates the common direction of its directed counterparts in $\vec{T}$), and the thick indecisive edges form a connected subtree. Note the following easy fact:

Fact A.4 *If e is a decisive edge with e' and e'' directed towards L_e, then $x_e = b_{in}(L_e) + b_{out}(L_e)$.*

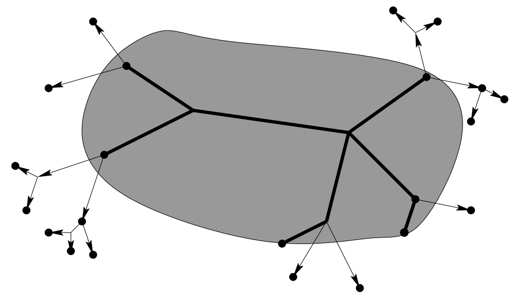

Figure 3: The structure of $AsymT$ solutions: the shaded area is the core.

Let us define a few useful terms: Let the *core* of T be the set of end-points of indecisive edges of T. Let $c(E') = \sum_{e \in E'} c_e$ for $E' \subseteq E$, and $St_T(V')$ be the minimal set of edges in T connecting vertices $V' \subseteq V$. Let $d_T(v, V')$ denote the minimum distance in T from v to a vertex in V', with respect to the edge-lengths c_e. Finally, for a subset $V' \subseteq V$, define

$$Q_T(V') = B\, c(St_T(V')) + \sum_{v \in W} (b_{in}(v) + b_{out}(v))\, d_T(v, V').$$

Claim A.5 *The cost of a tree solution T is $Q_T(core(T))$. Furthermore, for any subset V' of vertices in T, $Q_T(core(T)) \leq Q_T(V')$.*

Proof: The first part of the claim follows immediately from Claim A.1 and Fact A.4. For the second part, it is useful to break up $Q_T(V')$ into a weight assigned to each edge: each edge in $St_T(V')$ gets a weight of B. For $e \notin St_T(V')$, let L_e be the subtree of $T - e$ not containing any vertex of W, and assign weight $b_{in}(L_e) + b_{out}(L_e)$ to e. This weighted sum of the cost of edges in T, where each edge in T has been given a weight which is one of the four quantities in (A.6), is at most $Q_T(V')$. But $Q_T(core(T))$ is the same sum, where the weight in this case is equal to the minimum of those four quantities, which proves that $Q_T(core(T)) \leq Q_T(V')$. ∎

Let us give a procedure to find the optimal tree: find a tree T in V joining all demand points W, and a subset K of the vertices in T which minimizes the quantity $Q_T(K)$. To see that this is optimal, note that Claim A.5 implies that the cost of T is $Q_T(core(T))$, which is at most $Q_T(K)$. Consider the optimal solution T^*: it has cost $C(T^*) = Q_{T^*}(core(T^*))$, which is also of the form $Q_{\cdot}(\cdot)$. But since our procedure finds a pair T, K that minimizes that form, its cost is no more than $C(T^*)$, which proves the optimality.

For the final step of the reduction, let $St(K)$ denote the optimal Steiner tree in G connecting K, and for a vertex v, define $d_G(v, K)$ as the smallest distance from v to K in the graph G. Define $Q(K)$ as

$$Q(K) = B\, c(St(K)) + \sum_{v \in W} (b_{in}(v) + b_{out}(v))\, d_G(v, K).$$

We now show that finding the set K_{opt} minimizing $Q(K)$ (for $K \subseteq V$) is equivalent to solving $AsymT$, by showing that $Q(K_{opt}) = Q_{T^*}(K^*)$. Indeed, if V' is the set of vertices in $St_{T^*}(K^*)$, then $Q(V') \leq Q_{T^*}(V^*)$ and hence $Q(K_{opt}) \leq Q_{T^*}(V^*)$. Conversely, if T' is the optimal Steiner tree connecting K and K' is the node set of T', then $Q(V') \leq Q(K_{opt})$. Now for each node $v \in W$, we can add a shortest path to a node in V' such that the union of these paths and T' is still a tree, say T''. But now $Q_{T''}(V') \leq Q(K_{opt})$, and hence $Q_{T^*}(V^*) \leq Q(K_{opt})$, which completes the argument.

However, finding the set W is just an instance of CFL, where each $f_i = 0$, each $d_i = (b_{in}(v) + b_{out}(v))$ and $M = B$, and hence Theorem 2.6 implies a 9.002 approximation to the optimal tree solution.