# Simpler and Better Approximation Algorithms for Network Design

Anupam Gupta* Amit Kumar† Tim Roughgarden‡

## ABSTRACT

We give simple and easy-to-analyze randomized approximation algorithms for several well-studied NP-hard network design problems. Our algorithms improve over the previously best known approximation ratios. Our main results are the following.

- We give a randomized 3.55-approximation algorithm for the *connected facility location* problem. The algorithm requires three lines to state, one page to analyze, and improves the best-known performance guarantee for the problem.
- We give a 5.55-approximation algorithm for *virtual private network design*. Previously, constant-factor approximation algorithms were known only for special cases of this problem.
- We give a simple constant-factor approximation algorithm for the *single-sink buy-at-bulk network design* problem. Our performance guarantee improves over what was previously known, and is an order of magnitude improvement over previous combinatorial approximation algorithms for the problem.

## Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

## General Terms

Algorithms, Performance, Design, Theory

*Department of Computer Science, Carnegie Mellon University. This work was done while the author was visiting Lucent Bell Labs and was partly supported by a DIMACS grant. Email: anupamg@cs.cmu.edu.

†Lucent Bell Labs, 600 Mountain Avenue, Murray Hill NJ 07974. Email: amitk@research.bell-labs.com.

‡Department of Computer Science, Cornell University, Ithaca NY 14853. Supported by ONR grant N00014-98-1-0589. Email: timr@cs.cornell.edu.



*STOC'03,* June 9–11, 2003, San Diego, California, USA.



## Keywords

Network design, randomized algorithms, approximation algorithms

## 1. INTRODUCTION

We give simple and easy-to-analyze randomized approximation algorithms for three well-studied NP-hard network design problems. Our algorithms improve over the previously best known approximation ratios. We first give the definitions of these problems, deferring both their applications and related work to the end of the section.

**Connected Facility Location** (CFL). In the *connected facility location problem* (CFL), we are given an undirected graph $G = (V, E)$ with non-negative costs $c_e$ on edges, a set $D \subseteq V$ of *demands*, and a parameter $M > 1$. Each demand $j \in D$ has a non-negative *weight* $d_j$. A solution to an instance of CFL consists of a set $F \subseteq V$ of facilities to be opened, an assignment of demands to open facilities, and a subgraph $T$ of $G$ spanning $F$ (without loss of generality, $T$ is a tree). If such a solution assigns the demand $j$ to the open facility $i(j) \in F$, the cost of the solution is defined as $\sum_{j \in D} d_j \cdot \ell(j, i(j)) + M \sum_{e \in T} c_e$, where $\ell(\cdot,\cdot)$ denotes the shortest-path distance between two vertices in $G$ (w.r.t. edge lengths $c_e$). Connected facility location is thus the classical uncapacitated facility location problem, with no costs for opening facilities but with the additional constraint that open facilities must be connected together.

**Virtual Private Network Design** (VPND). In an instance of *Virtual Private Network design* (VPND) we are again given an undirected network with non-negative edge costs, a set $D \subseteq V$ of demands, and two non-negative *thresholds* $b_{in}(j)$ and $b_{out}(j)$ for each demand $j \in D$. These thresholds specify the maximum amount of traffic that demand $j$ will receive from and send to other demands. A $D \times D$ matrix describing the amount of (directed) traffic between each pair of demands is *valid* if it respects all thresholds. A feasible solution to an instance of VPND is given by a path $P_{ij}$ for each (ordered) demand pair $(i, j)$ and by capacities $u_e$ on edges sufficient to support all valid traffic matrices, with traffic from $i$ to $j$ routed on the path $P_{ij}$. The objective is to find a feasible solution minimizing the cost $\sum_{e \in E} c_e u_e$.

**Single-Sink Buy-at-Bulk Network Design** (SSBB). In an instance of *single-sink buy-at-bulk network design* (SSBB), we are given an undirected graph with non-negative edge costs, a *sink* vertex $t$, and a set $D \subseteq V$ of *demands*. We are also given $K$ types of *cables*, each specified by a capacity and a cost (per unit length). We seek a minimum-cost way of installing sufficient capacity on the edges so that a prescribed amount of flow can be sent simultaneously from each demand to the sink $t$. The cost per unit of capacity of a cable (the "bang for your buck") typically decreases as the cable cost increases, in accordance with *economies of scale*.

### *Our Results*

Our main results are the following.

1. We give a randomized approximation algorithm for CFL with a performance bound of $2+\rho_{ST}$, using a $\rho_{ST}$-approximation algorithm for the Steiner tree problem; the currently smallest available value for $\rho_{ST}$ is 1.55 [23]. This simple, intuitive and easily analyzed algorithm improves over the previously best known guarantee of $3+\rho_{ST}$, due to Swamy and Kumar [25].

2. We resolve the main open question posed in [13] by giving a 5.55-approximation algorithm for *virtual private network design*. Previously, constant-factor approximation algorithms were known only for special cases of this problem [9, 13]; the best known algorithm for the general case was a $O(\log n \log\log n)$ algorithm obtained by applying the tree embeddings of [6].

3. We give a simple constant-factor approximation algorithm for the *single-sink buy-at-bulk network design* problem. Our performance guarantee improves over what was previously known [26] by roughly a factor of 3, and gives an even-greater improvement over previous combinatorial approximation algorithms for the problem [12].

### *Related Work*

The connected facility location problem has received considerable recent attention both in the operations research literature [17, 19] and in the computer science community [13, 15, 16]. In addition to modeling the basic scenario of facility location in which some infrastructure among facilities must also be built, the problem naturally arises as a subroutine in several network design algorithms (see [13, 15]). Karger and Minkoff [15], motivated by the so-called *maybecast* problem, gave the first constant-factor approximation algorithm for the problem. This algorithm is simple and combinatorial, but has a relatively large performance guarantee. Gupta et al. [13] subsequently employed an LP-rounding approach to improve the approximation ratio. Very recently, Swamy and Kumar [25] discovered a primal-dual 4.55-approximation algorithm for the problem.

The virtual private network design problem considered in this paper was defined by Fingerhut et al. [9] and, subsequently and independently, by Duffield et al. [8]. It was later studied by Gupta et al. [13] with an eye toward approximation algorithms. Prior to the present work, constant-factor approximations were only known for restricted versions of VPND, such as the special case with $b_{in}(j) = b_{out}(j)$ for all demands $j$ [9, 13], and the case in which feasible solutions are restricted to route traffic on a tree [13].

Buy-at-bulk network design has been intensively studied from the viewpoint of approximation algorithms over the past few years. After the problem was introduced by Salman et al. [24], a long line of papers—that we will not review in detail here—have presented successively superior algorithms for increasingly general versions of the problem [1, 2, 6, 10, 11, 12, 18, 20, 21, 26]. For the SSBB problem considered here, the first nontrivial approximation was found by Awerbuch and Azar [2], using the tree embeddings of Bartal [5], and the first constant-factor approximation was given by Guha et al. [12]. The performance guarantee of the combinatorial algorithm of [12] was not stated explicitly, though Talwar [26] estimated it to be roughly 2000. Talwar [26] subsequently gave an LP-rounding algorithm with an improved performance guarantee of 216. For the special case of "access network design," Meyerson et al. [21] gave a simple randomized algorithm with a constant factor guarantee, but it is unclear how to extend the analysis of their algorithm to the more general case.

Finally, we note that many of these problems have also been studied in an online setting. Indeed, an online version of the algorithm of Section 2 is known to be $O(\log n)$-competitive for the so-called "rent-or-buy" problems [3, 4, 7], which are closely related to the CFL problem that we study here.[1] However, these techniques were not previously known to lead to constant-factor approximation algorithms for any of these offline problems.

### *Organization*

In Section 2 we describe our $(2+\rho_{ST})$-approximation algorithm for connected facility location. Building on the techniques used to analyze this algorithm, in Section 3 we give the first constant-factor approximation algorithm for VPN design. In Section 4 we present a simple 72.8-approximation algorithm for the single-sink buy-at-bulk network design problem.

## 2. CONNECTED FACILITY LOCATION

In this section, we present an intuitive and easy-to-implement randomized approximation algorithm for CFL with performance guarantee $2+\rho_{ST}$, using a $\rho_{ST}$-approximation algorithm for the Steiner tree problem. With the Steiner tree algorithm of Robins and Zelikovsky [23], we obtain a 3.55-approximation, improving upon the primal-dual 4.55-approximation algorithm of Swamy and Kumar [25].

We recall from Section 1 that in an instance of CFL, we are given an undirected graph $G = (V, E)$ with non-negative edge costs $c_e$, a set $D \subseteq V$ of demands, and a parameter $M > 1$. The objective is to identify a subset $F$ of the vertices $V$ as open facilities, and to build a Steiner tree $T$ connecting $F$ to minimize

$$\sum_{j \in D} d_j \cdot \ell(i(j), j) + M \cdot c(T),$$

where $i(j)$ is the closest open facility to demand $i$, $\ell$ is shortest-path distance (w.r.t. edge lengths $c_e$), $d_j$ is the weight of demand $j$, and $c(T)$ is the cost of the edges in the Steiner tree $T$. We will call the first term of the objective function the *connection cost*, and the second term the *Steiner cost*. We will refer to edges in the Steiner tree as *bought*, and edges in a shortest path between a demand $j$ and its nearest open facility $i(j)$ as *rented*.

We assume knowledge of a *root* facility $r \in V$ that is assuredly open in some optimal solution. This assumption is without loss of generality, since all $|V|$ "guesses" for a root $r$ can be tried one by one, with the best of all solutions obtained returned as output. We also assume for simplicity that $d_j = 1$ for all demands $j \in D$; this assumption is easy to remove, as we show at the end of the section. Let $C^*, S^*$ be the connection and Steiner costs of some optimal solution OPT that opens facility $r$. Let $Z^* = C^* + S^*$ denote the cost incurred by OPT, $F^* \subseteq V$ the facilities opened in OPT, and $T^*$ the Steiner tree on $F^*$ in OPT.

We now state our approximation algorithm for CFL. The algorithm can be viewed as a randomized reduction of CFL to the problem of finding a good Steiner tree, followed by the construction of a shortest-path tree.

[1] Precisely, the *single-sink rent-or-buy network design problem*, also known as the *network leasing problem*, is identical to connected facility location except for the additional constraint that a *root* vertex is required to be open in any feasible solution.

## 2.1 The Algorithm SIMPLECFL

C1. *Mark* each demand $j \in D$ with probability $1/M$, and let $D' \subseteq D$ denote the set of marked demands.

C2. Construct a $\rho_{ST}$-approximate Steiner tree on $F = D' \cup \{r\}$, and *buy* the edges of this tree.

C3. Assign each demand to its closest facility in $F$.

Our main theorem in this section is the following.

THEOREM 2.1. *The algorithm* SIMPLECFL *is a* $(2 + \rho_{ST})$-*approximation algorithm for* CFL.

The theorem will follow directly from the next two lemmas, which bound the expected Steiner cost and the expected connection cost separately.

LEMMA 2.2. *The expected cost of Step (C2) is at most* $\rho_{ST} \cdot Z^*$.

PROOF. It suffices to show that the expected cost of a min-cost Steiner tree on the (random) set of facilities $F$ is at most $Z^*$. We will prove this by using $T^*$, the Steiner tree on $F^*$ in OPT, to exhibit a (random) Steiner tree $T$ on $F$ with expected cost at most $Z^*$.

We define the Steiner tree $T$ on $F$ as the union of the edges of $T^*$ and the edges on shortest $j$-$i^*(j)$ paths for all $j \in F \setminus \{r\} \subseteq D$, where $j$ is assigned to $i^*(j)$ in OPT. The cost of $T^* \subseteq T$ is deterministically $S^*$. For a demand $j \in D$, the cost incurred for buying the shortest $j$-$i^*(j)$ path is $M \cdot \ell(j, i^*(j))$ with probability $1/M$ (if $j \in F$) and 0 otherwise (if $j \notin F$). In the worst case, all of the bought $j$-$i^*(j)$ shortest paths are edge-disjoint; linearity of expectation then implies the lemma:

$$\mathbf{E}\left[c(T)\right] \leq S^* + \sum_{j \in D} (1/M)\ M\ \ell(j, i^*(j)) = S^* + C^* = Z^*.$$

□

LEMMA 2.3. *The expected cost of Step (C3) is at most* $2 \cdot Z^*$.

PROOF. We first observe that the expected cost of the connections made in Step (C3) is independent of the particular Steiner tree constructed in Step (C2). We can therefore assume in our analysis, without loss of generality, that the Steiner tree of Step (C2) is given by the minimum spanning tree (in the graph of shortest-path distances) on $D$.

We now view the algorithm SIMPLECFL, employing the MST heuristic, in a new but essentially equivalent way. Instead of flipping coins for all demands at once, the new algorithm considers the demands one by one in some order and flips a coin for each in turn. Depending on the outcome of the coin flip, the demand is either (a) marked, added to $F$, and joined to the preexisting Steiner tree, or (b) connected to some previously marked vertex in $F$.

To decide the order on the vertices, we maintain two sets. At the beginning of step $t$, let $A_t$ be the set of vertices previously considered by the algorithm, and $B_t \subseteq A_t$ those that have been marked. Initially, $A_1 = B_1 = \{r\}$. In step $t$, we pick the vertex $v_t \in V \setminus A_t$ that is *closest* to $B_t$ and flip a coin for it. With probability $1/M$ (the outcome we call "heads"), we define $A_{t+1}$ and $B_{t+1}$ by adding $v_t$ to both the sets $A_t$ and $B_t$, and we update our Steiner tree by buying the shortest path from $v_t$ to its nearest neighbor in $B_t$. If the coin reads "tails", we set $A_{t+1} = A_t \cup \{v_t\}$ and $B_{t+1} = B_t$, and assign $v_t$ to its nearest neighbor in $B_t$.

A key observation is that the incremental process by which the Steiner tree $T$ is constructed on the marked facilities $F$ is nothing more than Prim's MST algorithm [22], running in the graph of shortest-path distances among vertices in $F$. Thus, this new randomized process faithfully implements the first two steps of SIMPLECFL. The connection cost incurred by this process is no less than that incurred by SIMPLECFL; we now complete the proof by showing that the expected connection cost for this new algorithm is at most $2Z^*$.

Let the random variable $X_t$ denote the cost from renting (assigning $v_t$ to its nearest neighbor in $B_t$) minus the cost of buying (adding $v_t$ to $B_t$ and connecting it to the existing Steiner tree) in step $t$ of the algorithm. Let $X = \sum_i X_i$ denote the connection cost minus the Steiner cost of the solution produced. The expected value of $X_t$, conditioning on the first $t-1$ coin flips so that $v_t$ and $B_t$ are deterministically known, is $(1 - 1/M) \cdot \ell(v_t, B_t) - (1/M) M \cdot \ell(v_t, B_t) \leq 0$. This inequality holds for any outcome of the first $t-1$ coin flips and hence holds unconditionally: $\mathbf{E}\left[X_t\right] \leq 0$ for all $t$. By linearity of expectation, $\mathbf{E}\left[X\right] \leq 0$ and the expected connection cost incurred by the incremental algorithm is at most the expected cost of the MST on $F$. The latter is at most $2Z^*$ by Lemma 2.2, since the MST heuristic is well known to 2-approximate the min-cost Steiner tree. The proof is complete. □

## 2.2 Extensions

Our analysis of algorithm SIMPLECFL is flexible and permits several extensions, as follows.

1. A naive way to allow non-uniform integral (or, by scaling, rational) demands, that will also be useful in later sections, is to modify Step (C1) so that $d_j$ coins are flipped for a demand $j$ with weight $d_j$; the demand is marked if at least one coin reads heads. Conceptually, we replace $j$ by $d_j$ co-located demands, each with weight 1. Since this is equivalent to flipping a coin for $j$ that comes up heads with probability $1-(1-1/M)^{d_j}$, this process can be implemented efficiently even when demand weights are not polynomially bounded.

   A simpler solution for connected facility location, that does not require integral demands, is to mark a demand $j \in D$ with probability $\min\{1, d_j/M\}$. Only cosmetic changes are required to generalize the proof of Theorem 2.1 to handle this modification.

2. The running time of the algorithm can be improved by a factor $|V|$ by choosing a root vertex $r$ uniformly at random from vertices in $D$. Modifying the above analysis gives a performance guarantee of $\alpha(2 + \rho_{ST})$ for this faster algorithm, where $\alpha = 1 + M/|D|$ is, without loss of generality, at most 2.

3. If facilities cannot be opened at arbitrary vertices of the graph (equivalently, facilities have costs that are either 0 or $+\infty$), relocating each demand to the nearest potential facility and running SIMPLECFL provides a $(4 + \rho_{ST})$-approximation. With general facility costs, a constant-factor approximation can be obtained from algorithm SIMPLECFL by computing an (approximate) Steiner Tree-Star [16, 25] instead of a Steiner tree in Step C2. The performance guarantee is slightly inferior to that of the 8.55-approximation algorithm of Swamy and Kumar [25], and the details of this reduction are omitted from this extended abstract.

# 3. VPN DESIGN

Motivated by the shortcomings of estimating or assuming knowledge of a fixed traffic matrix for a network, researchers proposed the problem of *virtual private network (VPN) design* [8, 9]. Recall from Section 1 that in this problem we are given *thresholds* $b_{in}(j)$ and $b_{out}(j)$ on the amount of traffic that enters and leaves a demand $j \in D \subseteq V$ of a network $G = (V, E)$ with edge costs $c_e$. The objective is to design a network which can handle *all* traffic patterns that respect the specified upper bounds. Formally, traffic is specified by a $D \times D$ matrix of non-negative real numbers, with entry $d_{ij}$ denoting the amount of traffic sent from demand $i$ to demand $j$. A traffic matrix is *valid* if the traffic incoming to any node $\sum_i d_{ij}$ is at most $b_{in}(j)$; also, the outgoing traffic $\sum_i d_{ji}$ should be bounded above by $b_{out}(j)$. We assume that thresholds are integral.

A solution to a VPND instance reserves bandwidth $u_e$ on edge $e$ in the graph, and fixes paths $P_{ij}$ between each ordered pair $i, j$ of demand nodes such that all valid traffic matrices can be routed using these paths without violating the reserved capacities. The cost of a solution is $\sum_e c_e u_e$ and we seek a solution of minimum cost.

In this section, we give a simple 5.55-approximation algorithm for this problem. Prior to our work, the best known solution was a straightforward application of Bartal's tree embeddings [6]; this approach only guarantees an $O(\log n \log \log n)$-approximation, where $n = |V|$ is the number of vertices. For the special case when $b_{in}(j) = b_{out}(j)$ for all demands $j$, a 2-approximation is known [9, 13], and Gupta et al. [13] gave a 10-approximation for the special case in which the the union of the routing paths $\{P_{ij}\}$ is required to form a tree.

Before stating our approximation algorithm, we make a couple of simplifying assumptions. By making many copies of each demand, we can assume that each demand $j$ is one of two types: a *sender* with $b_{in}(j) = 0$ and $b_{out}(j) = 1$, or a *receiver* with $b_{in}(j) = 1$ and $b_{out}(j) = 0$. As in Subsection 2.2, with a little more care this reduction can be efficiently implemented even when thresholds are not polynomially bounded. We will also assume that the receivers, $R$, outnumber the senders, $S$. The algorithm and analysis when $|R| \leq |S|$ is symmetric. We will let $M$ denote the number $|S|$ of senders.

The following algorithm, which we call SIMPLEVPN, builds a high-bandwidth "core" on one sender and a subset of the receivers, and routes all other senders and receivers to it using shortest paths.

V1. Choose a sender $s$ uniformly at random.

V2. Mark each receiver $j$ with probability $1/M$, and let $R'$ be the set of marked receivers.

V3. Construct a $\rho_{ST}$-approximate Steiner tree $T_s$ on $F = R' \cup \{s\}$; install capacity $M$ on all edges of $T_s$.

V4. For all senders and receivers $j$ not in the tree $T_s$, install one unit of capacity on the shortest path between $j$ and the set $F$.

In Step (V4), the effect of installing capacity on different shortest paths is cumulative; put differently, the capacity installed on an edge outside of $T_s$ is precisely the number of such shortest paths in which it is contained.

To begin the analysis, let $T$ denote the (random) set of edges that are assigned a nonzero capacity by SIMPLEVPN. With a consistent tie-breaking rule for shortest paths in Step (V4), $T$ will be a tree. The following lemma is straightforward.

LEMMA 3.1. *With probability 1, the tree $T$ produced by* SIMPLEVPN *is a feasible solution.*

We now bound the expected cost of the solution produced by algorithm SIMPLEVPN. We will do this by bounding three parts of the cost separately: the expected cost of Step (V3), the expected cost due to receivers in Step (V4), and the expected cost due to senders in Step (V4).

LEMMA 3.2. *The expected cost incurred in Step (V3) is at most $\rho_{ST} \cdot Z^*$, where $Z^*$ is the cost of an optimal* VPND *solution* OPT.

PROOF. We begin with an equivalent description of the random selection performed in Steps (V1) and (V2). Each receiver picks a sender uniformly at random, and we denote by $D_s$ the random set of demands picking sender $s$. We then pick a sender $s$ uniformly at random, and the Steiner tree instance of Step (V3) is then defined on $D_s \cup \{s\}$. The first two steps of SIMPLEVPN can be viewed as these same two (independent) selection steps, with the sender selected first and the random assignments of receivers to senders second (recall there are $M$ senders in all). To prove the lemma, it therefore suffices to prove that the expected cost of an optimal solution to a random Steiner tree instance on $D_s \cup \{s\}$ is at most $Z^*/M$. We will prove this inequality for an arbitrary fixed association of receivers to senders; the unconditional inequality then follows.

Fix a partition $\{D_s\}_{s \in S}$ of the receivers, and let $T_s^*$ denote a min-cost Steiner tree on $D_s \cup \{s\}$. Showing that $T_s^*$ has expected cost at most $Z^*/M$ (over the $M$ choices for $s$) is tantamount to proving that OPT can be "decomposed" into $M$ trees, each capable of handling any communication between a sender and its associated receivers:

$$\sum_s c(T_s^*) \leq Z^*.$$

To prove this inequality, first recall that the optimal solution OPT must specify a path $P_{rs}$ between each sender $s$ and receiver $r$. For a sender $s$, let $G_s$ be the subgraph $\cup_{r \in D_s} P_{rs}$. Since $G_s$ spans $D_s \cup \{s\}$, $c(T_s^*) \leq c(G_s)$. If edge $e$ appears in $k \geq 0$ subgraphs of the form $G_s$, then it is a member of $k$ sender-receiver paths that share no endpoints. Since simultaneous routing of traffic on these $k$ paths must be supported, OPT must install at least $k$ units of capacity on $e$. Therefore,

$$Z^* \geq \sum_s c(G_s) \geq \sum_s c(T_s^*),$$

which proves the lemma. □

The expected cost of joining the receivers to the central core in Step (V4) of SIMPLEVPN can be bounded above by $2Z^*$ in a manner identical to the proof Lemma 2.3; we omit further details.

LEMMA 3.3. *The expected cost incurred in Step (V4) from installing capacity on $r$-$F$ shortest paths for all receivers $r$ is at most $2Z^*$.*

Our final lemma bounds the expected cost of joining senders to the high-bandwidth core.

LEMMA 3.4. *The expected cost incurred in Step (V4) from installing capacity on $s'$-$F$ shortest paths for all senders $s'$ is at most $2Z^*$.*

PROOF. It suffices to show that, if a sender $s$ is picked uniformly at random, then

$$\mathbf{E}\left[\sum_{s' \in S} \ell(s, s')\right] \leq 2\, Z^*,$$

where $\ell(\cdot, \cdot)$ denotes shortest-path distance in $G$. To prove this inequality, we fix a set $R' \subseteq R$ of $M$ receivers. Any perfect matching $\mathcal{M}$ of $R'$ and $S$ naturally induces a valid traffic matrix that implies a lower bound of $\sum_{(r,s)\in\mathcal{M}} \ell(r,s)$ on $Z^*$. Averaging over all $M!$ possible perfect matchings, we obtain

$$\frac{1}{M}\sum_{r\in R', s\in S} \ell(r,s) \le Z^*,$$

since each receiver-sender pairing $(r,s)$ appears in $(M-1)!$ of the $M!$ perfect matchings. It follows from this inequality that

$$\mathbf{E}_{s\in S}\left[\sum_{r\in R'} \ell(r,s)\right] \le Z^*. \tag{3.1}$$

Also,

$$\begin{aligned}\sum_{s'\in S} \ell(s,s') \le \sum_{r\in R'} \ell(r,s) + \sum_{(r,s')\in\mathcal{M}} \ell(r,s') \\ \le \sum_{r\in R'} \ell(r,s) + Z^* \qquad (3.2)\end{aligned}$$

for an arbitrary perfect matching $\mathcal{M}$ of $R'$ and $S$. Combining (3.1) and (3.2) yields the lemma. $\square$

Combining Lemmas 3.2–3.4 with the Steiner tree algorithm of Robins and Zelikovsky [23] yields the main theorem of this section.

THEOREM 3.5. *Algorithm* SIMPLEVPN *is a 5.55-approximation algorithm for the VPN design problem.*

This resolves one of the main open questions from [13]. We had already noted that, assuming consistent tie-breaking, the solution output by SIMPLEVPN is a tree. Thus while tree solutions are not in general optimal for VPND [13], some tree solution is always near-optimal.

COROLLARY 3.6. *Every instance of* VPND *admits a tree solution with cost no more than 5.55 times that of an optimal (graph) solution.*

# 4. SINGLE SOURCE BUY-AT-BULK NETWORK DESIGN

In this section we give a simple constant-factor approximation algorithm for the widely studied SSBB problem. Our algorithm is based on that of Guha et al. [12], but our randomized techniques permit a simpler yet tighter analysis.

## 4.1 Notation and Preliminaries

Recall that in the SSBB problem we are given, in addition to the usual undirected network with edge costs, a *root* vertex $r$ and a set $D$ *demands*, with demand $j$ wishing to send $d_j$ units of flow to the root. As usual, we denote the length of an edge $e$ by $c_e$ and let $\ell$ denote shortest-path distance with respect to these lengths. Finally, there are $K$ *cable types* $\{1, 2, \ldots, K\}$, with the $i$th cable having capacity $u_i$ and cost $\sigma_i$ per cable per unit length. We define $\delta_i = \sigma_i/u_i$, which intuitively is the "incremental cost" of using cable type $i$. We will assume that each $u_i$ and $\sigma_i$ (and by definition $\delta_i$) is a power of 2. This assumption can be enforced while losing a factor of 4 in the approximation ratio (round each capacity $u_i$ down to the nearest power of 2, and each $\sigma_i$ up to the nearest power of 2).

We now note that costs and capacities must obey some geometric scaling properties. By reordering cable types, we can assume that $u_i < u_j$ and $\sigma_i < \sigma_j$ for all $i < j$. (If $u_i \le u_j$ and $\sigma_i \ge \sigma_j$, we can eliminate cable type $i$ from consideration.) Scaling, we can assume that $u_1 = \sigma_1 = 1$. The incremental costs $\delta_i$ then scale as well; note that

$$\sigma_k/u_k < \sigma_j/u_j \qquad \text{for each } j < k, \tag{4.3}$$

since otherwise we can eliminate cable type $k$ by replacing a cable of type $k$ by $u_k/u_j$ copies of type $j$ cables without increasing the cost. Since $\delta_j = \sigma_j/u_j$ is also a power of 2, this implies each $\delta_{j+1} \le \delta_j/2$ for all $j$, as $u_{j+1} \ge 2u_j$. Finally, we define $g_k = \frac{\sigma_{k+1}}{\sigma_k} u_k$; (4.3) implies that $g_k < u_{k+1}$, and hence

$$1 = u_1 < g_1 < u_2 < g_2 < \ldots < u_K < g_K = \infty. \tag{4.4}$$

Let OPT denote an optimal solution with cost $C^* = \sum_j C^*(j)$, where $C^*(j)$ is the amount paid for cables of type $j$.

We would like to assume that all demand weights are integral. This assumption is not without loss of generality, for we have already scaled cable capacities. Instead, we enforce this with the following "redistribution lemma". Roughly speaking, this lemma shows how to take a "grouping parameter" $U$ along with a tree with weights on its vertices, and randomly move weights throughout the tree so that the total weight at any node of the tree becomes either 0 or $U$. (For ensuring integral demands, we will take $U$ to be 1). Moreover, this random process has two important properties: the probability that a vertex in the tree receives weight $U$ is proportional to its initial weight, and no edge of the tree carries too much flow during the reallocation.

LEMMA 4.1 (**Redistribution Lemma**). *Let $T$ be a tree rooted at $r$ with each edge having capacity $U$. For each vertex $j \in T$, let $w(j) < U$ be the weight located at $j$ with $\sum_j w(j)$ a multiple of $U$. Then there is an efficiently computable (random) flow on the tree that redistributes weights without violating edge capacities, so that each vertex receives a new weight $w'(j)$ that is either 0 or $U$. Moreover,*

$$\mathbf{Pr}\left[\, j \text{ has } w'(j) > 0 \,\right] = w(j)/U \qquad \forall j. \tag{4.5}$$

A deterministic version of this lemma appears in [14, Lemma 1]. The proof is fairly simple, and we give it here only for the sake of completeness.

PROOF. Let us replace each edge in $T$ by two oppositely directed arcs. We first show that the lemma holds in this bidirected tree. First, we take an Euler tour of the vertices, yielding a cycle $C$. We also pick a value $Y$ drawn uniformly at random from $(0, U]$. We maintain a counter $Q$, which initially is set to 0.

We next go around the cycle, starting at the vertex $j_0 = r$, and visiting all the vertices $j_0, j_1, \ldots, j_m$ in (say) clockwise order. When we visit a vertex $j_k$, we set $Q \leftarrow Q + w(j_k)$. Suppose the counter $Q$, just before reaching $j_k$ was $Q_{old}$, and $Q_{new} = Q_{old} + w(j_k)$ is the value after accounting for $j_k$. If $xU + Y \in (Q_{old}, Q_{new}]$ for some integer $x$—i.e., the counter crossed the point $Y$ modulo $U$—then we "mark" $j_k$, and ask that it send $Q_{new} - (xU + Y)$ weight to the next marked vertex lying clockwise on the cycle. In the other case, we ask that the vertex send *all* its weight to the next marked vertex lying clockwise on the cycle. Note that the construction ensures that each arc on the cycle carries at most $U$ units of weight; furthermore, a vertex $j$ gets marked with probability $w(j)/U$, and this is exactly the probability that it has $U$ units of weight at the end of the process.

This process naturally induces a redistribution of weights in the original tree as well; however, since each edge of the tree was replaced by two arcs, there is a danger that the capacity of an edge may be violated by a factor of 2. This can be handled by rudimentary flow canceling. Let us consider an edge $e$ of the tree which was replaced by two opposite arcs $a$ and $\bar{a}$. Suppose both the arcs carry flow, with a path from $i$ to $j$ using $a$, and one from $i'$ to $j'$ using $\bar{a}$. We can now decrease the flow sent on these paths by $\epsilon$, and instead send $\epsilon$ flow from $i$ to $j'$, and from $i'$ to $j$. This does not change the amount of weight reaching a marked vertex, but decreases the total flow crossing $e$. This process stops when each edge is used in only one direction, at which point the flow crossing each edge of $T$ is at most $U$, completing the proof of the lemma. $\square$

With Lemma 4.1, we can build a $\rho_{ST}$-optimal Steiner tree $T_0$ with cables of capacity $u = u_1 = 1$ that connects all the demands, and use the procedure of Lemma 4.1 with $w(j)$ being the fractional part of $d_j$ to collect *integral* demands at some subset of vertices. The cost of the network to do this rerouting is just the cost of the Steiner tree built; since the tree built by the optimal solution is a candidate Steiner tree, we incur cost at most $\rho_{ST} \times \sum_j C^*(j)/\sigma_j$ (recall $\sigma_1 = 1$). Duplicating vertices if necessary, we can now assume that $d_j = 1$ for all $j$. We also assume that the number of demands $|D|$ is a power of 2; if not, we can place dummy demands at the root $r$ to achieve this.

## 4.2 The Algorithm SIMPLESSBB

The algorithm we present closely follows that of Guha et al. [12], where the network is designed incrementally in stages. At the $t$th stage, we use the value $u_{t+1}$ as an "aggregation threshold", and combine many demands (each of weight $u_t$) into a single demand of weight $u_{t+1}$. We buy cables on the paths required for this agglomeration. At the end of all these stages, the demand reaches the root; the path in the SSBB solution for this demand is then defined as the concatenation of the paths used in the aggregation stages.

To reiterate: at the beginning of the $t$th stage, there is a set $D_t$ of $|D|/u_t$ vertices with weight $u_t$, and other vertices have weight 0. (Initially $D_1 = D$, and we have enforced that each demand $j \in D$ has weight $d_j = 1 = u_1$.) The steps of stage $t$ are:

S1. *Mark* each demand in $D_t$ with probability $p_t = u_t/g_t$, and let $D'_t$ be the marked demands.

S2. Construct a $\rho_{ST}$-approximate Steiner tree $T_t$ on $F_t = D'_t \cup \{r\}$. Install a cable of type $(t+1)$ on each edge of this tree.

S3. For each vertex $j \in D_t$, sends its $u_t$ weight to the nearest member of $F_t$ using cables of type $t$. Let $w_t(i)$ be the weight collected at $i \in F_t$.

S4. A vertex $i \in F_t$ receives $u_t$ weight each from $w_t(i)/u_t$ vertices of $D_t$. Divide these vertices into groups of $u_{t+1}/u_t$ vertices each, leaving $b_i = (\frac{w_t(i)}{u_t} \bmod \frac{u_{t+1}}{u_t})$ *residual* vertices at the end. For each group of $u_{t+1}/u_t$ vertices, send the $u_{t+1}$ weight emanating from the group back from $i$ to a random member of the group, building new cables of type $t+1$ to do so.

After rerouting weight back to vertices of $D_t$ in this way for all $i \in F_t$, we use Lemma 4.1 with $T = T_t$, $w_t(i) = b_i u_t$ and $U = u_{t+1}$ to aggregate the weight from residual vertices into groups of weight exactly $u_{t+1}$. For every $i \in F_t$ that receives $u_{t+1}$ weight from this process, we send this weight back to one of $i$'s $b_i$ residual vertices, chosen uniformly at random, again building new cables of type $t+1$ to do so.

We note that $D_{t+1} \subseteq D_t$ for all $t$. Also, if $t = K$, then $p_K = 0$, so in the final iteration no demands are marked and all weight is sent to the root $r$ in Step (S3). We now analyze the algorithm with a sequence of simple lemmas.

LEMMA 4.2. *For every non-root vertex $j \in D$ and stage $t$,*

$$\mathbf{Pr}\left[\, j \in D_t \,\right] = 1/u_t .$$

PROOF. The proof is by induction. The claim is clearly true for $t = 1$. For stage $t$, consider $j \in D_t$. Suppose $j$ sends its weight to vertex $i \in F_t$ in Step (S3). The vertex $j$ is either a residual vertex of $i$, or it is not. In the former case, Lemma 4.1 assigns $u_{t+1}$ weight to $i$ with probability $b_i u_t/u_{t+1}$, and $j$ subsequently receives this weight with probability $1/b_i$. In the latter case, $j$ receives the group of $u_{t+1}$ weight collected at $i$ to which it belongs with probability $u_t/u_{t+1}$. In either case,

$$\mathbf{Pr}\left[\, j \in D_{t+1} \,\right] = \mathbf{Pr}\left[\, j \in D_{t+1} \,|\, j \in D_t \,\right] \mathbf{Pr}\left[\, j \in D_t \,\right] = (u_t/u_{t+1})(1/u_t) = 1/u_{t+1}.$$

$\square$

As a corollary of this result, we get that a non-root vertex lies in $F_t$ with probability $p_t \times 1/u_t = 1/g_t$.

LEMMA 4.3. *Let $T_t^*$ be the optimal Steiner tree on $F_t$, and $c(T_t^*) = \sum_{e \in T_t^*} c_e$. Then*

$$\mathbf{E}\left[\, c(T_t^*) \,\right] \;\leq\; \sum_{s>t} \frac{1}{\sigma_s}\, C^*(s) + \sum_{s \leq t} \frac{1}{\delta_s \cdot g_t}\, C^*(s). \qquad (4.6)$$

PROOF. We assume for simplicity that all demand weights were initially 1 (i.e., that Lemma 4.1 was not needed as a preprocessing step); the general case requires only a mildly more complicated argument.

We will exhibit a (random) graph $G_t$ spanning $F_t$ that has low expected cost. We first add to $G_t$ all the edges in OPT possessing a cable of type $t+1$ or higher. If $E_s$ is the set of edges with a cable of type $s$, then $c(E_s) \leq C^*(s)/\sigma_s$, which gives the first summation of (4.6).

We complete $G_t$ by considering each vertex $i \in F_t \setminus \{r\} \subseteq D$ in turn. In OPT, demand $i$ may use several $i$-$r$ paths to send flow to the root $r$. (We unfortunately cannot assume without loss of generality that OPT is a tree). We randomly add to $G_t$ one of these paths, with a path chosen with probability equal to the fraction of $i$'s weight that it carries.

Consider an edge $e$ of $G$ with no cable of type $t+1$ or higher. Suppose for simplicity that only one cable is installed on $e$, say of type $s \leq t$. Then $e$ lies in $G_t$ if and only if, for some $i \in D$ and some $i$-$r$ flow path $P$ containing $e$ in OPT, $i$ was selected for $F_t$ and then $P$ was selected among all $i$-$r$ flow paths. Since each $i \in D$ lies in $F_t$ with probability $1/g_t$, it follows from the Union Bound that $e$ lies in $G_t$ with probability at most $f_e/g_t$, where $f_e$ is the amount of flow on $e$ in OPT. Since $f_e \leq u_s$, edge $e$ contributes at most $c_e u_s/g_t$ to the expected cost of $G_t$. On the other hand, this cable of type $s$ on edge $e$ contributes $\sigma_s c_e$ to $C^*(s)$. Thus the expected cost in $G_t$ for edge $e$ is $1/(g_t\,\delta_s)$ times what OPT pays for the cable. For edges on which multiple cables are installed, this same analysis can be performed on a cable-by-cable basis. Summing over all edges with no cable of type $t+1$ or higher now proves the lemma. $\square$

We now relate the cost of our algorithm to the cost of this random Steiner tree on $F_t$.

LEMMA 4.4. *The expected cost incurred in stage $t$ is at most $(3 + \rho_{ST})\,\sigma_{t+1}\,\mathbf{E}\left[\, c(T_t^*) \,\right]$, where $T_t^*$ is the optimal Steiner tree on $F_t$.*

PROOF. The cost of the Steiner tree in Step (S2) is at most $\rho_{ST}\,\sigma_{t+1}\,c(T_t^*)$, while the cost of Step (S3) is at most $2\,\sigma_{t+1}\,c(T_t^*)$ by an argument analogous to that proving Lemma 2.3.

We complete the proof by bounding the cost of redistributing the flow back to randomly chosen demands in Step (S4). Lemma 4.1 ensures that the rerouting of weight from residual vertices can be accomplished using the cables of type $t+1$ purchased in Step (S2), and no new cables need be built. A group of $u_{t+1}$ weight at a vertex of $F_t$ is returned to one the $u_{t+1}/u_t$ vertices of $D_t$ from which the weight emanated. Since the vertex is chosen at random, the expected cost for the cables of type $t+1$ supporting this return trip is the cost incurred for this weight in Step (S3), times $\delta_{t+1}/\delta_t$. Summing over all groups and using the scaling properties of incremental costs, the total cost of the cables for routing flow back to random vertices of $D_t$ is at most $(1/2)\cdot 2\,\sigma_{t+1}\,c(T_t^*) \le \sigma_{t+1}\,c(T_t^*)$, and the lemma is proved. □

THEOREM 4.5. *Algorithm* SIMPLESSBB *is a 72.8-approximation algorithm for the* SSBB *problem.*

PROOF. In our preprocessing step, we incur a factor 4 loss from rounding the costs and the capacities to powers of 2. Furthermore, we incur a cost of $\rho_{ST}\sum_j C^*(j)/\sigma_j$ to ensure that we have integral demands at each vertex. The cost incurred during the algorithm proper is obtained by plugging (4.6) into the statement of Lemma 4.4, and summing over all $t$. This shows that the coefficient of $C^*(s)$ is at most

$$4\times(3+\rho_{ST})\times\left(\sum_{t=0}^{s-1}\sigma_{t+1}/\sigma_s+\sum_{t\ge s}\delta_t/\delta_s\right). \qquad (4.7)$$

Since the $\sigma_t$ and $\delta_t$ are powers of 2, both sums are geometric and are bounded above by 2. This implies a guarantee of $(3+\rho_{ST})\times 4\times 4$, which is at most 72.8. □

## 5. CONCLUSIONS

We have exhibited several simple randomized approximation algorithms for network design problems that improve over the previously best known performance guarantees. There are several natural questions that demand further study. Can our algorithms be derandomized? Can the analysis of this paper be improved, or are there tight examples demonstrating that our analysis of these algorithms is best possible? Is the randomized framework of this paper sufficiently powerful to tackle harder network design problems?

## 6. REFERENCES


[1] Matthew Andrews and Lisa Zhang. Approximation algorithms for access network design. *Algorithmica*, 34(2):197–215, 2002. (Preliminary version in 39th FOCS, 1998.).

[2] 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.

[3] Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized Steiner problem. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (Atlanta, GA, 1996)*, pages 68–74, New York, 1996. ACM.

[4] Yair Bartal. *Competitive Analysis of Distributed On-line Problems — Distributed Paging*. PhD thesis, Tel-Aviv University, Israel, 1994.

[5] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 184–193, 1996.

[6] Yair Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 161–168, 1998.

[7] Yair Bartal, Amos Fiat, and Yuval Rabani. Competitive algorithms for distributed data management. *J. Comput. System Sci.*, 51(3):341–358, 1995. (Preliminary version in 24th STOC, 1992).

[8] 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.

[9] J. Andrew Fingerhut, Subhash Suri, and Jonathan S. Turner. Designing least-cost nonblocking broadband networks. *J. Algorithms*, 24(2):287–309, 1997.

[10] Naveen Garg, Rohit Khandekar, Goran Konjevod, R. Ravi, F. Sibel Salman, and Amitabh Sinha. On the integrality gap of a natural formulation of the single-sink buy-at-bulk network design formulation. In *Proceedings of the 8th IPCO*, LNCS vol 2081, pages 170–184, 2001.

[11] 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.

[12] Sudipto Guha, Adam Meyerson, and Kamesh Mungala. A constant factor approximation for the single sink edge installation problems. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing (STOC)*, pages 383–388, 2001.

[13] Anupam Gupta, Amit Kumar, Jon Kleinberg, Rajeev Rastogi, and Bülent Yener. Provisioning a Virtual Private Network: A network design problem for multicommodity flow. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 389–398, 2001.

[14] Refael Hassin, R. Ravi, and F. S. Salman. Approximation algorithms for a capacitated network design problem. In *APPROX*, pages 167–176, 2000.

[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] Samir Khuller and An Zhu. The general Steiner tree-star problem. *Information Processing Letters*, 84:215–220, 2002.

[17] Tae Ung Kim, Timothy J. Lowe, Arie Tamir, and James E. Ward. On the location of a tree-shaped facility. *Networks*, 28(3):167–175, 1996.

[18] Amit Kumar, Anupam Gupta, and Tim Roughgarden. A constant-factor approximation algorithm for the multicommodity rent-or-buy problem. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pages 333–342, 2002.

[19] Youngho Lee, Yuping Chiu, and Jennifer Ryan. A branch and cut algorithm for a Steiner tree-star problem. *INFORMS Journal on Computing*, 8(3):194–201, 1996.

[20] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Cost-distance: Two metric network design. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 624–630, 2000.

[21] Adam Meyerson, Kamesh Munagala, and Serge Plotkin. Designing networks incrementally. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 406–415, 2001.

[22] Robert C. Prim. Shortest interconnection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[23] 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.

[24] F. Sibel Salman, Joseph Cheriyan, R. Ravi, and Sairam Subramanian. Approximating the single-sink link-installation problem in network design. *SIAM Journal on Optimization*, 11(3):595–610, 2000.

[25] Chaitanya Swamy and Amit Kumar. Primal-dual algorithms for the connected facility location problem. In *Proceedings of the 5th APPROX*, LNCS vol 2462, pages 256–269, 2002.

[26] Kunal Talwar. Single-sink buy-at-bulk LP has constant integrality gap. In *Proceedings of the 9th IPCO*, LNCS vol 2337, pages 475–486, 2002.