Approximation via Cost Sharing: Simpler and Better Approximation Algorithms for Network Design

ANUPAM GUPTA
Carnegie Mellon University
AMIT KUMAR
IIT Delhi
MARTIN PÁL
Google, Inc.
and
TIM ROUGHGARDEN
Stanford University

We present constant-factor approximation algorithms for several widely-studied NP-hard optimization problems in network design, including the multicommodity rent-or-buy, virtual private network design, and single-sink buy-at-bulk problems. Our algorithms are simple and their approximation ratios improve over those previously known, in some cases by orders of magnitude.

We develop a general analysis framework to bound the approximation ratios of our algorithms. This framework is based on a novel connection between random sampling and game-theoretic cost sharing.

Categories and Subject Descriptors: F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Nonnumerical Algorithms and Problems*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Approximation algorithms, cost sharing, network design, random sampling

---

The work of A. Gupta was supported in part by NSF CAREER award CCF-0448095 and an Alfred P. Sloan Fellowship, and was performed in part while the author was at Lucent Bell Laboratories. The work of A. Kumar was performed in part while the author was at Cornell University and Lucent Bell Laboratories. The work of M. Pál was supported by ONR grant N00014-98-1-0589 and performed in part while the author was at Cornell University. The work of T. Roughgarden was supported by in part by ONR grant N00014-04-1-0725, DARPA grant W911NF-04-9-0001, and an NSF CAREER Award, and was performed in part while the author was at Cornell University. Preliminary versions of these results appeared in the Proceedings of the 35th Annual Symposium on Theory of Computing, June 2003 and the Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science, October 2003.

Authors' addresses: A. Gupta, Department of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213, email anupamg@cs.cmu.edu. A. Kumar, Department of Computer Science and Engineering, IIT Delhi, India 110016, email amitk@cse.iitd.ernet.in. M. Pál, Google, Inc., 76 9th Avenue, 4th Floor New York, NY 10011, mpal@acm.org. T. Roughgarden, Department of Computer Science, Stanford University, 462 Gates Building, Stanford CA 94305, email tim@cs.stanford.edu.

1. INTRODUCTION

We present constant-factor approximation algorithms for several widely-studied NP-hard optimization problems in network design. Our algorithms are extremely simple and have the following flavor: randomly sample a simpler subproblem, solve the subproblem with an existing algorithm, and greedily extend the subproblem solution to a solution feasible for the original problem. The approximation ratios of our algorithms improve over all of those previously known, in some cases by orders of magnitude.

We develop a general analysis framework to bound the approximation ratios of our algorithms. This framework is based on a novel connection between random sampling and *cost sharing*, the task of allocating the cost of an object to many users of the object in a "fair" manner. Specifically, we define the notion of *strict* cost shares, and show that such cost shares provide a powerful tool for analyzing the performance of a class of random sampling algorithms.

1.1 Three Network Design Problems

To describe our results more concretely, we define the three primary network design problems that we consider in this paper. We discuss the motivation for and prior work on these problems in Section 1.3 below.

Problem 1.1 *Multicommodity Rent-or-Buy.* An instance of the *multicommodity rent-or-buy* (MRoB) problem is defined by an undirected graph $G = (V, E)$ and a set $\mathcal{D} = \{(s_i, t_i)\}_{i=1}^{k}$ of vertex pairs called *demand pairs*, where each edge $e \in E$ has a nonnegative length c_e and each demand pair (s_i, t_i) has a nonnegative weight w_i. The goal is to compute a minimum-cost way of installing sufficient capacity on the edges so that w_i units of flow can be sent simultaneously from each source s_i to the corresponding sink t_i. The cost of installing capacity on an edge is given by a simple concave function: capacity can be *rented*, with cost incurred on a per-unit of capacity basis, or *bought*, which allows unlimited use after payment of a large fixed cost. Precisely, there are positive parameters μ and M, with the cost of renting capacity equal to μ times the capacity required (per unit length), and the cost of buying infinite capacity equal to M (per unit length). By scaling, we can assume that $\mu = 1$ without loss of generality. The cost of a solution can thus be expressed as

$$\sum_{e \in E_b} c_e M + \sum_{e \in E_r} c_e u_e,$$

where E_b denotes the edges on which capacity is bought and E_r the rest of the edges, and where u_e denotes the amount of capacity rented on the edge $e \in E_r$. We denote an MRoB instance by a tuple $(G, \mathcal{D}, w, M)$ and leave the length vector c implicit.

We will also study the special case of *single-sink rent-or-buy* (SSRoB), where all demand pairs (s_i, t_i) share a common sink vertex t, and the more general *multicast rent-or-buy* problem (MuRoB), where there are arbitrary demand groups instead of demand pairs.

Problem 1.2 *Virtual Private Network Design.* In an instance of *virtual private network design* (VPND), we are again given an undirected graph G with nonnegative

edge lengths c. There is also a set D of *demands*, each of which is located at a vertex of G. Each demand $j \in D$ possesses two nonnegative *thresholds* $b_{in}(j)$ and $b_{out}(j)$. These thresholds specify the maximum amount of traffic that demand j will receive from and send to other demands, respectively. 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 specified by a path P_{ij} for each ordered demand pair (i, j) and a capacity u_e for each edge e, such that there is sufficient capacity to route every valid traffic matrix via the paths $\{P_{ij}\}$. These paths are allowed to form cycles. The objective is to find a feasible solution that minimizes the cost $\sum_{e \in E} c_e u_e$. We denote an instance of VPND by the triple (G, D, b).

Problem 1.3 *Single-Sink Buy-at-Bulk.* The *single-sink buy-at-bulk network design* (SSBaB) problem is a generalization of the SSRoB problem. The input is the same as in the latter problem, except that instead of a single parameter M describing the cost of buying capacity, there are K types of *cables*. A cable of type i has a given capacity u_i and a given cost (per unit length) σ_i. As in the SSRoB problem, the goal is to compute a minimum-cost way of installing sufficient capacity on the edges so that a prescribed amount of flow w_i can be sent simultaneously from each source s_i to the common sink t. There is no limit on the number of cables or the number of types of cables that can be installed on each edge.

The SSBaB problem also has a well-known alternative formulation, in which the cost of installing capacity (per-unit length) is governed by a fixed concave function. From an approximation viewpoint, these two formulations are equivalent up to a factor of 2. The SSRoB problem is clearly a special case of the second formulation.

The following simpler network design problem arises frequently as a subroutine in our algorithms.

Problem 1.4 *Steiner Forest.* An instance of the Steiner Forest problem is given by an undirected graph G with nonnegative edge lengths c and a set $\mathcal{D} = \{(s_i, t_i)\}_{i=1}^{k}$ of demand pairs. The goal is to compute a minimum-cost subgraph of G that contains an s_i-t_i path for every $i \in \{1, 2, \ldots, k\}$, where the cost of a subgraph is the sum of the lengths of its edges. We denote such a Steiner Forest instance by $(G, \mathcal{D})$.

The Steiner Forest problem is equivalent to the special case of the MRoB problem in which $M = 1$ and $w_i \geq 1$ for every i. If every demand pair of a Steiner Forest instance has a common sink, then it is equivalent to an instance of the well-known Steiner Tree problem. All of the problems studied in this paper contain Steiner Tree as a special case.

Recall that an *α-approximation algorithm* for a minimization problem runs in polynomial time and returns a solution of cost no more than α times that of an optimal solution. The value α is the *approximation ratio* or *performance guarantee* of the algorithm. Since even the Steiner Tree problem is MAX-SNP-hard [Bern and Plassman 1989], Problems 1.1–1.3 cannot be solved exactly or approximated to within an arbitrarily small constant factor in polynomial time, assuming $P \neq NP$ [Arora et al. 1998]. We are therefore justified in seeking constant-factor approximation algorithms for these problems, with the constant as small as possible.

Problem	Previously Best Approximation	This Paper
MRoB	over 1000 [Kumar et al. 2002]	small constant
MuRoB	$O(\log n)$ [Awerbuch and Azar 1997; Fakcharoenphol et al. 2004]	small constant
SSRoB	4.55 [Swamy and Kumar 2004]	3.55
VPND	$O(\log n)$ [Fakcharoenphol et al. 2004; Gupta et al. 2001]	5.55
SSBaB	216 [Talwar 2002]	76.8

Table I. Main results of this paper. "Previously best approximation" refers to the smallest approximation ratio known prior to the conference versions of our work [Gupta et al. 2003; Gupta et al. 2003]. The parameter n denotes the number of network vertices. For the MRoB and MuRoB problems, a number of different papers have used the algorithmic and analytic framework of this paper to give successively better approximation ratios; the current records are 5 and 6, respectively [Fleischer et al. 2006].

1.2 Overview of Results

Our main results are the following.

—We develop an analysis framework that shows that random sampling, a Steiner Forest subroutine, and greedy augmentation leads to a constant-factor approximation algorithm for the MRoB problem, provided the subroutine admits what we call strict cost shares (defined in Section 2). Beginning with the conference version of this work [Gupta et al. 2003], a number of authors have provided strict cost-sharing methods for several different Steiner Forest algorithms, culminating in a recent 5-approximation algorithm for the MRoB problem due to Fleischer et al. [2006]. We also extend our analysis framework to the MuRoB problem.

—For the SSRoB problem, we show that every Steiner Tree algorithm admits strict cost shares and obtain a randomized 3.55-approximation algorithm.

—For the VPND problem, we build on our SSRoB algorithm and analysis to obtain a randomized 5.55-approximation algorithm.

—We combine ideas from our SSRoB algorithm and analysis with an SSBaB algorithm of Guha et al. [2001] to obtain a randomized 76.8-approximation algorithm for the SSBaB problem.

Prior to our work, the best-known approximation ratios for the MRoB, MuRoB, SSRoB, VPND, and SSBaB problems were over 1000 [Kumar et al. 2002]; $O(\log n)$, where n is the number of network vertices [Awerbuch and Azar 1997; Fakcharoenphol et al. 2004]; 4.55 [Swamy and Kumar 2004]; $O(\log n)$ (which follows simply from the structural results in [Gupta et al. 2001] combined with the tree approximations from [Fakcharoenphol et al. 2004]); and 216 [Talwar 2002], respectively. See also Table I. Our constant-factor approximation algorithm for the VPND problem answers the main open questions of Gupta et al. [2001]. We note that all the previous best-known algorithms can be made deterministic, whereas the algorithms presented in this paper are randomized.

Finally, our approximation algorithm for MRoB gives qualitatively new information about the relative tractability of different network design problems with economies of scale. Specifically, for many years even the simplest such problems with multiple commodities (like MRoB) seemed more difficult than relatively complex single-sink network design problems (such as SSBaB). While a constant-factor

approximation algorithm for MRoB was devised in [Kumar et al. 2002], its complexity and extremely large approximation ratio did little to change this perception. The MRoB algorithm of this paper shows that this state of affairs arose only because of a lack of a good algorithm for MRoB, not because of the problem's intrinsic difficulty.

1.3 Related Work

The literature on approximation algorithms for NP-hard network design problems is vast, and we will only discuss work that is directly related to the problems studied in this paper. In this subsection, we mainly discuss research that occurred prior to or independent of the present work. Since the publication of preliminary versions of the results in this paper [Gupta et al. 2003; Gupta et al. 2003], there has been much research on further applications, generalizations, and improvements of our algorithms and analysis techniques. We survey this recent research in Section 6.

1.3.1 *Rent-or-Buy Network Design.* Rent-or-buy problems have long served as a simple model of network design with economies of scale—where the per-unit cost of installing capacity on an edge decreases as more capacity is installed. They also arise naturally in other applications, including stochastic optimization problems [Karger and Minkoff 2000] and facility location problems [Kumar et al. 2002; Swamy and Kumar 2004].

For many years, the best algorithm known for the MRoB problem was an $O(\log n \log\log n)$-approximation algorithm, where n denotes the number of network vertices, due to Awerbuch and Azar [1997] and Bartal [1998]. (Recent work by Fakcharoenphol et al. [2004] can be used to improve the approximation ratio of this algorithm to $O(\log n)$.) The first constant-factor approximation algorithm for the problem is due to Kumar et al. [2002]. However, both the analysis and the primal-dual algorithm of [Kumar et al. 2002] are quite complicated, and the performance guarantee shown for the algorithm is, while constant, extremely large. This constant was neither optimized nor estimated in [Kumar et al. 2002], but it is at least 1000. Our MRoB algorithm is the first constant-factor approximation algorithm for the problem that is simple or that has a reasonably small constant performance guarantee.

The SSRoB special case of MRoB, and the closely related *connected facility location* problem, have been extensively studied in the operations research literature [Kim et al. 1996; Labbé et al. 2001; Lee et al. 1996] and by the computer science community [Gupta et al. 2001; Karger and Minkoff 2000; Khuller and Zhu 2002; Ravi and Salman 1999; Swamy and Kumar 2004]. The first constant-factor approximation for the problem was given by Ravi and Salman [1999], who studied it as the *traveling purchaser problem* and gave an LP-rounding algorithm. Independently of their work, Karger and Minkoff [2000] gave another constant-factor approximation algorithm for the problem, motivated by the so-called *maybecast* problem. This algorithm is simple and combinatorial, but has a relatively large performance guarantee. Gupta et al. [2001] essentially rediscovered the LP-rounding approach of Ravi and Salman [1999]; their paper improved on the approximation ratio. Prior to our work, the best algorithm for the problem was the primal-dual 4.55-approximation algorithm due to Swamy and Kumar [2004].

Finally, our random sampling approach to the MRoB problem is reminiscent of

and partially inspired by previous work that gave online algorithms with polylogarithmic competitive ratios for many rent-or-buy-type problems [Awerbuch et al. 2004; Bartal 1994; Bartal et al. 2001; Bartal et al. 1995].

1.3.2 *Virtual Private Network Design.* The virtual private network design problem considered in this paper was defined by Fingerhut et al. [1997] and, subsequently and independently, by Duffield et al. [1999]. The model is motivated by the many difficulties in estimating or assuming knowledge of a fixed traffic matrix for a network (see [Duffield et al. 1999; Fingerhut et al. 1997]). The VPND problem was later studied by Gupta et al. [2001] with an eye toward approximation algorithms.

Prior to our work, the best known algorithm for the VPND problem was a direct application of probabilistic tree embeddings [Fakcharoenphol et al. 2004], which only guarantees a $O(\log n)$-approximation, where n is the number of vertices. For the special case of VPND where $b_{in}(j) = b_{out}(j)$ for every demand $j \in D$, a 2-approximation is known [Fingerhut et al. 1997; Gupta et al. 2001]. Also, Gupta et al. [2001] gave a 10-approximation algorithm for the special case of the VPND problem in which the the union of the routing paths $\{P_{ij}\}_{i,j \in D}$ is required to form a tree. They also showed that this case is a special case of the SSRoB problem, and hence the results mentioned above for the SSRoB problem also apply to it.

1.3.3 *Buy-at-Bulk Network Design.* Rent-or-buy problems are a special case of *buy-at-bulk network design*, where the goal is the same but the cost of installing capacity is given by an arbitrary concave function (or, nearly equivalently, by a set of cable types). Buy-at-bulk network design has been intensely studied over the last several years. After the problem was introduced by Salman et al. [2000], a long line of papers have presented successively superior algorithms for increasingly general versions of the problem.

For the SSBaB problem considered here (Problem 1.3), the first non-trivial approximation was found by Awerbuch and Azar [1997], using the tree embeddings of Bartal [1996], and the first constant-factor approximation algorithm was given by Guha et al. [2001]. The performance guarantee of the combinatorial algorithm in [Guha et al. 2001] was not stated explicitly, though Talwar [2002] estimated it to be roughly 2000. Talwar [2002] subsequently gave an LP-rounding algorithm with an improved performance guarantee of 216, the best known before our work.

Many researchers have studied other types of single-sink network design problems with economies of scale, including the more specialized Access Network Design problem [Andrews and Zhang 2002; Guha et al. 2000; 2001; Meyerson et al. 2001], and the generalizations of SSBaB in which the capacity cost function can be edge-dependent [Chekuri et al. 2001; Meyerson et al. 2000] or unknown to the algorithm [Goel and Estrin 2005]. The best known approximation ratios for these three problems are 68 [Meyerson et al. 2001], $O(\log n)$ [Chekuri et al. 2001; Meyerson et al. 2000], and $O(\log n)$ [Goel and Estrin 2005], respectively. Recent results of Chuzhoy et al. [2005] rule out constant-factor approximation algorithms for the second problem under reasonable complexity-theoretic assumptions.

For the multicommodity buy-at-bulk network design problem, the best known approximation ratio is $O(\log n)$, which follows from combining the algorithm of Awerbuch and Azar [1997] with the probabilistic tree embeddings given by Fakcharoen-

phol et al. [2004]. Andrews [2004] recently proved that, under reasonable complexity-theoretic assumptions, there is no constant-factor approximation algorithm for this problem. Very recently, Charikar and Karagiozova [2005], Chekuri et al. [2006], and Gupta et al. [2006] developed the first non-trivial approximation algorithms for various generalizations of the multicommodity buy-at-bulk network design problem.

1.3.4 *Steiner Forest.* The first non-trivial approximation algorithm for the Steiner Forest problem was the 2-approximation algorithm due to Agrawal et al. [1995]. Subsequently, Goemans and Williamson [1995; 1997] reinterpreted the algorithm and analysis of [Agrawal et al. 1995], and generalized them to a wide class of network design problems. Recently, Könemann et al. [2005] gave a somewhat different 2-approximation algorithm for the Steiner Forest problem.

1.3.5 *Cost Sharing.* Cost sharing has long been a fundamental subject in game theory and economics; see e.g. [Young 1994] and the references therein. Our definition of strict cost-sharing methods in Section 2 is somewhat reminiscent of well-known concepts in cooperative game theory, including the *core* and the *nucleolus*. However, we are not aware of any work in the game theory literature that studies our notion of strict cost sharing.

There has long been an informal connection between primal-dual approximation algorithms for network design and cost sharing. Many cost-minimization network design problems admit a natural integer programming formulation, and the dual linear program of the linear relaxation of this integer program can usually be interpreted as approximately sharing the cost of an optimal solution (see e.g. [Goemans and Williamson 1997]). In addition, techniques from approximation algorithms have recently yielded new progress on several cost-sharing problems [Gupta et al. 2004; Jain and Vazirani 2001; 2002; Könemann et al. 2005; Pál and Tardos 2003]. Despite these previous connections, we believe the present work to be the first to show that the explicit use of cost-sharing methods can lead to better approximation algorithms, and the first to use cost sharing to analyze random sampling algorithms.

1.4 Paper Organization

Section 2 presents our analysis framework, defines strict cost shares, and proves that random sampling, a Steiner Forest subroutine that admits strict cost shares, and greedy augmentation leads to a constant-factor approximation algorithm for MRoB. Section 3 applies this framework to the SSRoB, MRoB, and MuRoB problems. In Section 4, we build on our SSRoB algorithm and analysis and design a constant-factor approximation algorithm for the VPND problem. Section 5 applies our analysis tools to the SSBaB problem. Sections 3–5 all logically depend on the concepts in Section 2. Sections 4 and 5 also depend on Section 3.1, though Sections 3–5 are otherwise independent. Finally, Section 6 discusses recent work motivated by this paper and possible directions for future research.

2. THE ANALYSIS FRAMEWORK

This section describes our high-level algorithm and analysis framework for the MRoB problem. Section 2.1 presents our MRoB algorithm. Section 2.2 bounds its expected cost when solving a randomly sampled subproblem. Section 2.3 defines strict cost shares, and Section 2.4 uses them to bound the expected cost of

Input: an MRoB instance $(G, \mathcal{D}, w, M)$.

(1) (*Sampling step*) Choose a random subset $\mathcal{S} \subseteq \mathcal{D}$ of demand pairs, by including each pair $(s_i, t_i) \in \mathcal{D}$ in $\mathcal{S}$ independently with probability $\min\{w_i/M, 1\}$.
(2) (*Subproblem step*) Compute a feasible solution F to the Steiner Forest instance $(G, \mathcal{S})$, and buy (infinite) capacity on the edges of F.
(3) (*Augmentation step*) Greedily rent capacity to produce a feasible solution.

Fig. 1. The algorithm SAMPLE-AUGMENT.

the greedy augmentation step of our MRoB algorithm.

2.1 Random Sampling and Greedy Augmentation

Our algorithm for the MRoB problem is given in Figure 1. It first randomly samples a subset of demand pairs, with probabilities proportional to weights and inversely proportional to the ratio M of the buying and renting costs. It then buys capacity on edges so that each demand pair in the random sample is connected by an infinite-capacity path. Finally, our algorithm augments the capacity of the bought edges by greedily renting capacity for all demand pairs that did not participate in the random sample.

The sampling step in Figure 1 is self-explanatory. For the subproblem step, we will employ an algorithm that is a good approximation algorithm for Steiner Forest and also satisfies an additional property that we describe in Section 2.3. We implement the augmentation step as follows. After the subproblem step, every demand pair (s_i, t_i) in the subset $\mathcal{S}$ is connected by a path of (infinite-capacity) bought edges in F. Let G/F denote the graph obtained from G by contracting all of the edges of F. Independently for each demand pair $(s_i, t_i) \notin \mathcal{S}$, we compute a shortest s_i-t_i path $\widehat{P}_i$ of G/F, and rent w_i units of capacity on each edge of $\widehat{P}_i$ that are reserved for exclusive use by (s_i, t_i). Each path $\widehat{P}_i$ corresponds to an s_i-t_i path P_i of G, where each edge of P_i either has infinite capacity or has w_i units of capacity reserved for the demand pair (s_i, t_i). The augmentation step thus installs sufficient capacity for all of the demand pairs to simultaneously route their traffic on the paths $\{P_i\}_{i=1}^{k}$.

The following lemma will be used in the next subsection and also motivates the SAMPLE-AUGMENT algorithm.

LEMMA 2.1. *For every* MRoB *instance, there is an optimal solution such that the flow of each demand pair can be routed on a single path.*

PROOF. Fix an arbitrary MRoB instance $(G, \mathcal{D}, w, M)$ and an optimal solution for it. Let F denote the edges on which the optimal solution buys infinite capacity. This optimal solution must also, independently for each demand pair (s_i, t_i), reserve w_i units of capacity on s_i-t_i paths of the contracted graph G/F. The minimum-cost way to accomplish this is to rent w_i units of capacity for each demand pair (s_i, t_i) on a shortest s_i-t_i path of G/F, as in the augmentation step of the SAMPLE-AUGMENT algorithm. Applying this augmentation step to the set F thus results in an optimal solution in which the traffic of each demand pair can be routed on a single path. □

The proof of Lemma 2.1 shows that the augmentation step of the algorithm SAMPLE-

AUGMENT extends the subproblem solution to a feasible solution in an optimal way. The crux of the MRoB problem is to identify a good set of edges on which to buy infinite capacity. We will show that the random Steiner Forest instance defined by the sampling step of the SAMPLE-AUGMENT algorithm leads to such a good set of edges.

The rest of this section is devoted to proving that, provided the right type of Steiner Forest algorithm is used in the subproblem step, the algorithm SAMPLE-AUGMENT is a good approximation algorithm for MRoB. In Section 3 we discuss Steiner Forest algorithms that possess the requisite properties.

2.2 Bounding the Subproblem Cost

Algorithm SAMPLE-AUGMENT incurs cost both in the subproblem step (for buying capacity) and in the augmentation step (for renting capacity). We first prove a key lemma that is useful for bounding both of these costs. The lemma states that, in expectation, there is a low-cost solution to the random Steiner Forest instance solved in the subproblem step of the algorithm SAMPLE-AUGMENT.

LEMMA 2.2. *For every instance $\mathcal{I} = (G, \mathcal{D}, w, M)$ of* MRoB*,*

$$\mathbf{E}\left[OPT_{\mathcal{S}}\right] \leq \frac{OPT_{MRoB}}{M}, \tag{1}$$

where OPT_{MRoB} is the cost of an optimal solution for $\mathcal{I}$, $OPT_{\mathcal{S}}$ is the cost of an optimal solution for the Steiner Forest *instance $(G, \mathcal{S})$, and the expectation is over the random sample $\mathcal{S}$ chosen in the sampling step of the algorithm* SAMPLE-AUGMENT.

PROOF. Fix an instance $\mathcal{I}$ of MRoB. We prove (1) by exhibiting one feasible solution for each possible Steiner Forest instance $(G, \mathcal{S})$, such that the expected cost (over $\mathcal{S}$) of this solution is at most OPT_{MRoB}/M. Since this goal is only for the analysis, and is independent of the algorithm SAMPLE-AUGMENT, we can freely make use of an optimal solution for $\mathcal{I}$. By Lemma 2.1, we can consider an optimal solution that routes all of the traffic of each demand pair $(s_i, t_i) \in \mathcal{D}$ on a single path P_i^*. For an edge e, let $x_e^* = \sum_{i\,:\,e \in P_i^*} w_i$ denote the amount flow routed on the edge e. Let E_b denote the edges e with $x_e^* \geq M$ and E_r the rest of the edges. The cost OPT_{MRoB} of the optimal solution is

$$OPT_{MRoB} = \sum_{e \in E_b} c_e M + \sum_{e \in E_r} c_e x_e^*. \tag{2}$$

To prove (1), fix a possible random sample $\mathcal{S} \subseteq \mathcal{D}$, and define a Steiner forest $F_{\mathcal{S}}$ by

$$F_{\mathcal{S}} = E_b \cup \bigcup_{(s_i,t_i) \in \mathcal{S}} P_i^*.$$

Note that $F_{\mathcal{S}}$ consists of one part (E_b) that does not depend on $\mathcal{S}$, and one part ($\cup_{(s_i,t_i)\in\mathcal{S}} P_i^*$) that does, and is certainly a feasible solution for the Steiner Forest instance $(G, \mathcal{S})$. The cost of the first part is deterministically $c(E_b) = \sum_{e \in E_b} c_e$, a factor of M less than the cost incurred by the optimal solution for $\mathcal{I}$ for buying capacity on these edges. The expected cost of the second part is a factor of M

less than the cost incurred by the optimal solution for renting capacity, because we include a demand pair (s_i, t_i) in the sample $\mathcal{S}$ with probability only w_i/M. Formally, we bound the expected cost of $F_{\mathcal{S}}$ as follows:

$$\begin{aligned}
\mathbf{E}\left[c(F_{\mathcal{S}})\right] &= \mathbf{E}\left[c(E_b)\right] + \mathbf{E}\left[c\left(E_r \cap \left(\cup_{(s_i,t_i)\in\mathcal{S}} P_i^*\right)\right)\right] \\
&= c(E_b) + \sum_{e\in E_r} c_e \cdot \mathbf{Pr}\left[e \in \cup_{(s_i,t_i)\in\mathcal{S}} P_i^*\right] \\
&\leq c(E_b) + \sum_{e\in E_r} c_e \sum_{i\,:\,e\in P_i^*} \mathbf{Pr}[(s_i,t_i)\in\mathcal{S}] \\
&= c(E_b) + \sum_{e\in E_r} c_e \frac{x_e^*}{M},
\end{aligned}$$

where the inequality follows from the Union Bound. Thus the expected cost of $F_{\mathcal{S}}$ is at most the cost of an optimal solution (2) divided by M. Since $\mathbf{E}\left[OPT_{\mathcal{S}}\right] \leq \mathbf{E}\left[c(F_{\mathcal{S}})\right]$, this proves the lemma. □

Lemma 2.2 easily implies that the expected cost of the subproblem step of Sample-Augment is small provided a good approximation algorithm for Steiner Forest is used.

Lemma 2.3. *If an α-approximation algorithm for* Steiner Forest *is used in the subproblem step of* Sample-Augment*, then the expected cost incurred in this step is at most α times the cost of an optimal* MRoB *solution.*

Proof. Fix an arbitrary instance $\mathcal{I}$ of MRoB. Let $\mathcal{A}$ be the α-approximation algorithm used in the subproblem step of Sample-Augment. The cost incurred in this step is M times that of the Steiner forest F returned by $\mathcal{A}$, since Sample-Augment buys infinite capacity on the edges of F. This cost is at most $M\cdot\alpha\cdot OPT_{\mathcal{S}}$ for every possible random sample $\mathcal{S}$ of demand pairs. The expected cost is thus at most $M \cdot \alpha \cdot \mathbf{E}\left[OPT_{\mathcal{S}}\right]$, which by Lemma 2.2 is at most $\alpha \cdot OPT_{MRoB}$. □

The next two subsections undertake the more challenging task of bounding the expected cost of the augmentation step of the Sample-Augment algorithm.

2.3 Strict Cost Shares

Our analysis of the expected cost of the augmentation step of the Sample-Augment algorithm hinges on a type of cost sharing for the Steiner Forest problem. We next define what we call *strict cost shares*. While our definition is motivated solely by our analysis of Sample-Augment, it can also be interpreted as formalizing a natural approximate fairness condition.

The next definition states that a cost-sharing method is a way of allocating cost to the demand pairs of a Steiner Forest instance $(G, \mathcal{D})$, with the total cost allocated bounded above by that of an optimal Steiner forest for $(G, \mathcal{D})$.

Definition 2.4. Let χ be a function that, for every instance $\mathcal{I} = (G, \mathcal{D})$ of Steiner Forest, assigns a nonnegative real value $\chi(\mathcal{I}, (s_i, t_i))$ to every demand pair $(s_i, t_i) \in \mathcal{D}$. The function χ is a *(Steiner forest) cost-sharing method* if, for every such

instance $\mathcal{I}$,

$$\sum_{(s_i,t_i)\in\mathcal{D}} \chi(\mathcal{I},(s_i,t_i)) \leq OPT(\mathcal{I}), \tag{3}$$

where $OPT(\mathcal{I})$ is the cost of an optimal solution to $\mathcal{I}$.

Definition 2.4 permits some rather uninteresting cost-sharing methods, including the function that always assigns all demand pairs zero cost. The key additional property that we require is that, intuitively, a cost-sharing method allocates each demand pair a cost share commensurate with its distance from the edges needed to connect all of the other demand pairs. Put differently, no demand pair can be a "free rider," imposing a large burden in building a Steiner forest, but only receiving a small cost share. We call cost-sharing methods with this property *strict*. Strict cost shares will allow us to charge, in a demand pair-by-demand pair fashion, a constant fraction of the expected cost of the augmentation step of Sample-Augment to the expected cost of an optimal solution to the Steiner Forest subproblem. We have already bounded the latter cost in Lemma 2.2.

To make this idea precise, we require further notation. Let $\ell_G(u,v)$ denote the length of a shortest path between the vertices u and v in the graph G (with respect to the edge lengths of G). As in Section 2.1, for a graph G and a set of edges F, G/F denotes the graph obtained from G by contracting all of the edges of F. As in the augmentation step of the algorithm Sample-Augment, the minimum per-unit cost of renting capacity between s_i and t_i, given that infinite capacity has already been bought on the edges in F, is precisely $\ell_{G/F}(s_i,t_i)$. Our main definition is then the following.

Definition 2.5. Let $\mathcal{A}$ be a deterministic algorithm for the Steiner Forest problem. A Steiner forest cost-sharing method χ is *β-strict for $\mathcal{A}$* if for all instances $\mathcal{I} = (G,\mathcal{D})$ and for all demand pairs $(s_i,t_i)\in\mathcal{D}$,

$$\ell_{G/F}(s_i,t_i) \leq \beta\cdot\chi(\mathcal{I},(s_i,t_i)),$$

where F is the Steiner forest returned for the instance $(G,\mathcal{D}\setminus\{(s_i,t_i)\})$ by the algorithm $\mathcal{A}$.

Remark 2.6. Definition 2.4 makes no reference to an algorithm for Steiner Forest, but Definition 2.5 does. Thus a Steiner forest cost-sharing method can be β-strict for one algorithm and not for another. For example, every cost-sharing method is strict with respect to the (highly suboptimal) algorithm that always returns the entire graph G as the Steiner forest solution F. Our challenge will be to give a strict cost-sharing method for a good approximation algorithm for Steiner Forest.

We say that an algorithm is strict if it admits a strict cost-sharing method.

Definition 2.7. An algorithm $\mathcal{A}$ for the Steiner Forest problem is *β-strict* if there exists a cost-sharing method that is β-strict for $\mathcal{A}$.

Strict cost shares will pay dividends in Lemma 2.9 below, where we use them to bound the expected augmentation cost of the algorithm Sample-Augment.

Example 2.8 *Prim Cost Shares.* We now give a strict cost-sharing method for the special case of the SSRoB problem, where all demand pairs share the same

sink vertex t. In this case, the subproblem step is an instance $(G, \mathcal{S})$ of Steiner Tree, where we must output a set F of edges spanning t and all of the source vertices s_i in demand pairs of $\mathcal{S}$. Suppose we use the well-known MST heuristic as our Steiner Tree algorithm $\mathcal{A}$, implemented with Prim's MST algorithm (see e.g. [Vazirani 2001]). In more detail, we iteratively build up a feasible solution to $(G, \mathcal{S})$ as follows. Initially, set $D = \{t\}$ and $F = \emptyset$. At each iteration, among all sources in a demand pair of $\mathcal{S}$ but not in D, find the source s_i closest to some source or sink already in D; add s_i to D; and add to F a shortest path between s_i and its nearest neighbor in D.

For an instance $\mathcal{I} = (G, \mathcal{S})$ of Steiner Tree, define the cost share $\chi(\mathcal{I}, (s_i, t))$ of (s_i, t) as half of the length of the shortest path used in the iteration of the algorithm that adds s_i to D. We call these *Prim cost shares*. We claim that the function χ satisfies both Definition 2.4 and Definition 2.5 with $\beta = 2$. Definition 2.4 is met because the sum of all of the cost shares is exactly half of the cost of the Steiner tree output by the MST heuristic, which in turn is at most twice the cost of an optimal Steiner tree (see e.g. [Vazirani 2001]).

To see why the cost shares χ are 2-strict for the algorithm $\mathcal{A}$, consider an arbitrary Steiner Tree instance $\mathcal{I} = (G, \mathcal{S})$ and demand pair $(s_i, t) \in \mathcal{S}$. Consider running the algorithm $\mathcal{A}$ in parallel on the instances $\mathcal{I}$ and $\widehat{\mathcal{I}} = (G, \mathcal{S} \setminus \{(s_i, t)\})$. A key observation is that these two executions of $\mathcal{A}$ are identical, until the demand pair (s_i, t) of $\mathcal{I}$ is considered. In other words, if $\mathcal{A}$ chooses (s_i, t) in iteration $j \geq 1$ of its execution for the original instance $\mathcal{I}$, then the partial solution F_{j-1} that $\mathcal{A}$ has constructed after $j - 1$ iterations is the same in both executions of the algorithm. (We will see in Section 3.1 that this is a much stronger property than is needed for the analysis.)

Suppose when algorithm $\mathcal{A}$ is run on the instance $\mathcal{I}$, it connects s_i to F_{j-1} via the path P, where P is a shortest path between s_i and some previously added source or sink. Since $\mathcal{A}$'s final solution $\widehat{F}$ to the instance $\widehat{\mathcal{I}}$ includes F_{j-1}, the shortest-path distance $\ell_{G/\widehat{F}}(s_i, t)$ is at most the cost $c(P)$ of P. Since the cost share $\chi(\mathcal{I}, (s_i, t))$ is precisely $c(P)/2$, Definition 2.5 is satisfied with $\beta = 2$.

2.4 Bounding the Augmentation Cost

The definition of strict cost shares is engineered so that the following upper bound on the expected augmentation cost of the algorithm Sample-Augment holds.

Lemma 2.9. *If a β-strict algorithm for* Steiner Forest *is used in the subproblem step of* Sample-Augment*, then the expected cost incurred in the augmentation step of* Sample-Augment *is at most β times the cost of an optimal* MRoB *solution.*

Proof. Suppose the β-strict Steiner Forest algorithm $\mathcal{A}$ is used in the subproblem step of the algorithm Sample-Augment and fix an MRoB instance $(G, \mathcal{D}, w, M)$. For each demand pair $(s_i, t_i) \in \mathcal{D}$, we define two random variables. First, the random variable R_i ("renting cost") has value 0 if (s_i, t_i) is included in the random sample $\mathcal{S}$, and otherwise has value equal to the renting cost $w_i \cdot \ell_{G/F}(s_i, t_i)$ caused by (s_i, t_i) in the augmentation step, where F is the Steiner Forest solution returned by $\mathcal{A}$ for the instance $(G, \mathcal{S})$. Second, the random variable B_i ("buying cost") has value $M \cdot \chi((G, \mathcal{S}), (s_i, t_i))$ if (s_i, t_i) is included in the random sample $\mathcal{S}$ and 0 otherwise. Note that the cost incurred by Sample-Augment in the augmentation

step is precisely the total renting cost $\sum_i R_i$. With probability 1, the total buying cost satisfies

$$\sum_{i=1}^{k} B_i = \sum_{(s_i,t_i)\in\mathcal{S}} M \cdot \chi((G,\mathcal{S}),(s_i,t_i)) \leq M \cdot OPT_{\mathcal{S}}, \tag{4}$$

where the inequality follows from Definition 2.4. Lemma 2.2 then implies that the expected total buying cost is at most the cost OPT_{MRoB} of an optimal solution to $(G,\mathcal{D},w,M)$:

$$\mathbf{E}\left[\sum_{i=1}^{k} B_i\right] \leq OPT_{MRoB}. \tag{5}$$

The rest of the proof shows how to use strict cost shares to charge, up to a factor of β, the expected renting cost incurred by Sample-Augment to the expected buying cost.

Fix a demand pair (s_i,t_i). Condition on the set $\mathcal{S} \subseteq \mathcal{D} \setminus \{(s_i,t_i)\}$ of other demand pairs that Sample-Augment includes in its random sample. Let $\widehat{\mathcal{S}}$ denote $\mathcal{S}\cup\{(s_i,t_i)\}$. Thus the subproblem step will involve either the Steiner Forest instance $\widehat{\mathcal{I}} = (G,\widehat{\mathcal{S}})$ (with probability $\min\{w_i/M,1\}$) or the instance $\mathcal{I} = (G,\mathcal{S})$ (with the remaining probability). The expected renting cost incurred by (s_i,t_i), conditioned on $\mathcal{S}$, can therefore be crudely bounded by

$$\mathbf{E}\left[R_i|\mathcal{S}\right] = \left(1 - \min\left\{\frac{w_i}{M},1\right\}\right)\cdot w_i \cdot \ell_{G/F}(s_i,t_i) \leq \min\{w_i,M\}\cdot \ell_{G/F}(s_i,t_i), \tag{6}$$

where F is the output of $\mathcal{A}$ for the Steiner Forest instance $\mathcal{I}$. The expected buying cost is

$$\mathbf{E}\left[B_i|\mathcal{S}\right] = \min\left\{\frac{w_i}{M},1\right\}\cdot M \cdot \chi(\widehat{\mathcal{I}},(s_i,t_i)) = \min\{w_i,M\}\cdot\chi(\widehat{\mathcal{I}},(s_i,t_i)). \tag{7}$$

Strict cost shares provide the key relation between renting and buying costs. Specifically, since $\mathcal{A}$ is β-strict, inequality (6) and equation (7) imply that

$$\mathbf{E}\left[R_i|\mathcal{S}\right] \leq \beta \cdot \mathbf{E}\left[B_i|\mathcal{S}\right].$$

Since this inequality holds for every set $\mathcal{S} \subseteq \mathcal{D}\setminus\{(s_i,t_i)\}$, it also holds unconditionally:

$$\mathbf{E}\left[R_i\right] \leq \beta \cdot \mathbf{E}\left[B_i\right].$$

Linearity of expectation and inequality (5) complete the proof:

$$\begin{aligned}\mathbf{E}\left[\sum_{i=1}^{k} R_i\right] &\leq \beta \cdot \mathbf{E}\left[\sum_{i=1}^{k} B_i\right] \\ &\leq \beta \cdot OPT_{MRoB}.\end{aligned} \tag{8}$$

□

Lemmas 2.3 and 2.9 immediately imply the main result of this section: Sample-Augment is a good approximation algorithm for MRoB, provided a good, strict Steiner Forest algorithm is used in the subproblem step.

THEOREM 2.10. *If a β-strict α-approximation algorithm for* Steiner Forest *is used in the subproblem step of* SAMPLE-AUGMENT*, then* SAMPLE-AUGMENT *is a randomized $(\alpha+\beta)$-approximation algorithm for* MRoB*.*

3. RENT-OR-BUY PROBLEMS

We next apply the analysis framework of Section 2, and Theorem 2.10 in particular, to several rent-or-buy problems. We begin in Section 3.1 with the special case of the SSRoB problem, and show how the results of Section 2 easily give a simple algorithm with a better performance guarantee than all previously known approximation algorithms for the problem. Section 3.2 considers the MRoB problem, and motivates and surveys several constant-factor approximation algorithms that have appeared since the conference version of the present work [Gupta et al. 2003]. Finally, Section 3.3 treats the MuRoB problem.

3.1 Single-Sink Rent-or-Buy

A good approximation algorithm for the SSRoB problem follows immediately from the Prim cost shares of Example 2.8 and Theorem 2.10. Specifically, in Example 2.8 we argued that the MST heuristic is a 2-approximation algorithm for the Steiner Tree problem and admits 2-strict cost shares. Theorem 2.10 then implies the following.

THEOREM 3.1. *Algorithm* SAMPLE-AUGMENT*, with the subproblem step implemented with the MST heuristic, is a 4-approximation algorithm for the* SSRoB *problem.*

Theorem 3.1 already improves over the previously best algorithm for the SSRoB problem, the primal-dual 4.55-approximation algorithm of Swamy and Kumar [2004].

We can achieve a slightly better approximation ratio by refining Definition 2.5 and Theorem 2.10 for the SSRoB problem. For the rest of this subsection, we call a source or sink of a Steiner Tree instance a *demand*. Let D be the set of all demands; hence $D = \cup_{\{s_i,t\}\in\mathcal{D}}\{s_i\} \cup \{t\}$.

Definition 3.2. A Steiner tree cost-sharing method χ is *universally β-strict* if for all Steiner Tree instances $\mathcal{I} = (G, \mathcal{D})$ and for all demand pairs $(s_i, t) \in \mathcal{D}$,

$$\ell(s_i, D \setminus \{s_i\}) \leq \beta \cdot \chi(\mathcal{I}, (s_i, t)),$$

where D denotes the set of demands of $\mathcal{I}$ and $\ell(s_i, D\setminus\{s_i\})$ the length of a shortest path between s_i and some other demand.

Example 3.3. Recall that the Prim cost shares defined in Example 2.8 assign to each demand pair (s_i, t) a cost share equal to half of the length of a shortest path between s_i and some other demand. This is at least half of the length $\ell(s_i, D\setminus\{s_i\})$ of the shortest such path. Prim cost shares are therefore universally 2-strict.

The next lemma justifies the use of the word "universal" in Definition 3.2: universally strict cost shares are strict with respect to every Steiner Tree algorithm.

LEMMA 3.4. *If χ is a universally β-strict Steiner tree cost-sharing method and $\mathcal{A}$ is a* Steiner Tree *algorithm, then χ is β-strict for $\mathcal{A}$.*

PROOF. To satisfy Definition 2.5, we must show that $\ell_{G/F}(s_i,t) \leq \beta \cdot \chi(\mathcal{I},(s_i,t))$ for every Steiner Tree instance $\mathcal{I} = (G,\mathcal{D})$ and every demand pair $(s_i,t) \in \mathcal{D}$, where F is the output of $\mathcal{A}$ for the Steiner Tree instance $(G, \mathcal{D} \setminus \{(s_i,t)\})$. Letting D denote the set of demands of $\mathcal{I}$, we have as

$$\ell_{G/F}(s_i,t) \leq \ell(s_i, D \setminus \{s_i\}) \leq \beta \cdot \chi(\mathcal{I},(s_i,t)),$$

where the first inequality follows from the fact that F includes a path between t and every demand in $D \setminus \{s_i\}$, and the second inequality follows from Definition 3.2. □

Theorem 2.10 and Lemma 3.4 immediately give the following result.

THEOREM 3.5. *Suppose there is a universally β-strict Steiner tree cost sharing method. If an α-approximation algorithm for* Steiner Tree *is used in the subproblem step of* SAMPLE-AUGMENT*, then* SAMPLE-AUGMENT *is a randomized $(\alpha+\beta)$-approximation algorithm for* SSRoB.

Theorem 3.5 decouples the tasks for finding a good Steiner Tree approximation algorithm and finding (universally) strict cost shares. Combining the universally 2-strict Prim cost shares and the 1.55-approximation algorithm for Steiner Tree due to Robins and Zelikovsky [2005] then yields a 3.55-approximation algorithm for SSRoB.

COROLLARY 3.6. *There is a randomized 3.55-approximation algorithm for the* SSRoB *problem.*

Remark 3.7. The same graphs that show that the MST heuristic is no better than a 2-approximation algorithm for Steiner Tree prove that for every constant $\beta < 2$, there is no universally β-strict Steiner tree cost sharing method. (Consider a sink and a large number of sources at distance 2 from each other, and an additional node at distance $1+\epsilon$ from each of the others [Vazirani 2001, Example 3.4].) On the other hand, better upper bounds on the approximation ratio of SAMPLE-AUGMENT could follow from stricter cost shares that are not universally strict, or from improvements to the upper bound in Theorem 3.5. For example, the approximation factor in Corollary 3.6 can be slightly improved by choosing the sampling rate in the SAMPLE-AUGMENT algorithm more carefully.

Remark 3.8. In the proof of Lemma 3.4, we crucially used the fact that every feasible solution to a Steiner Tree instance includes all of the demands in a single connected component. Since different feasible solutions to a Steiner Forest instance can connect the demands in fundamentally different ways, there do not seem to be useful analogues of Definition 3.2 and Theorem 3.5 for the Steiner Forest and MRoB problems, respectively.

3.2 Multicommodity Rent-or-Buy

We now consider the MRoB problem. Theorem 2.10 reduces the problem of designing a constant-factor approximation algorithm to the problem to designing an $O(1)$-strict cost-sharing method for an $O(1)$-approximation algorithm for the Steiner Forest problem. Several papers, beginning with the conference version of this work [Gupta et al. 2003], have designed such cost-sharing methods. Since the analysis in [Gupta et al. 2003] is complicated and the approximation factor achieved

Fig. 2. Example 3.9. A Steiner Forest instance showing that no local cost-sharing method for the AKR-GW algorithm is $O(1)$-strict.

there has been subsumed by more recent work, we restrict ourselves to motivating and surveying this line of research. For details on the analysis in [Gupta et al. 2003], see Pál [2004].

3.2.1 *The* AKR-GW *Algorithm and a Motivating Example.* Until recently [Könemann et al. 2005], the only known $O(1)$-approximation algorithm for the Steiner Forest problem was the 2-approximation algorithm developed by Agrawal et al. [1995] and generalized by Goemans and Williamson [1995]. We refer to this algorithm as the AKR-GW algorithm. We briefly review this algorithm in the Appendix.

A natural idea is to use the AKR-GW algorithm as the Steiner Forest subroutine in the Sample-Augment algorithm and attempt to define $O(1)$-strict cost shares for it. By Theorem 2.10, such cost shares would give a constant-factor approximation algorithm for MRoB. Moreover, since the AKR-GW algorithm is a primal-dual algorithm (see the Appendix), the dual variables constructed by the AKR-GW algorithm naturally suggest Steiner forest cost-sharing methods. In particular, when a dual variable y_S is increased by an additive factor of Δ, we could increase the cost shares of the demand pairs that are separated by S by at most Δ, with this increase split between these cost shares in an arbitrary way. The sum of cost shares defined in this way is at most the value of the dual solution constructed by the AKR-GW algorithm, which in turn is at most the value of an optimal Steiner forest. Such cost shares thus satisfy Definition 2.4. But are they strict?

Our next example shows that no cost-sharing scheme of this type is $O(1)$-strict for the AKR-GW algorithm. Precisely, call a Steiner forest cost-sharing method χ *local for* AKR-GW if, for every Steiner Forest instance $\mathcal{I} = (G, \mathcal{D})$ and every demand pair $(s_i, t_i) \in \mathcal{D}$, the cost share $\chi(\mathcal{I}, (s_i, t_i))$ is at most the sum of the dual variables y_S of the AKR-GW algorithm that correspond to clusters S that separate (s_i, t_i). Note that all of the cost-sharing methods in the aforementioned family are local for AKR-GW in this sense.

Example 3.9. Consider the Steiner Forest instance $\mathcal{I}$ shown in Figure 2, where n is arbitrarily large and $\epsilon < 1/n$. We will show that every cost-sharing method χ that is local for AKR-GW is $\Omega(n)$-strict for AKR-GW.

Consider the execution of the AKR-GW algorithm on the instance $\mathcal{I}$ just after the time $\frac{1}{2}$. There are $n+1$ clusters: s_1 and t_1 are each in an (active) singleton cluster, and s_i and t_i share an (inactive) cluster for $i = 2, 3, \ldots, n$. By the time $\tau^* = (1+\epsilon n)/2$, all of the vertices lie in the same (inactive) cluster. The maximum cost share that can be allocated to the demand pair (s_1, t_1) by the local cost-sharing

method χ is $2\tau^* = O(1)$.

Now let $\widehat{\mathcal{I}}$ denote the instance $(G, \mathcal{D} \setminus \{(s_1, t_1)\})$ and consider the execution of the AKR-GW algorithm on $\widehat{\mathcal{I}}$. All clusters are inactive by the time $\frac{1}{2}$, and the final output of the algorithm is the set F of unit length edges. The s_1-t_1 distance $\ell_{G/F}(s_1, t_1)$ is thus $n(1+\epsilon) = \Omega(n)$. The cost-sharing method χ is therefore $\Omega(n)$-strict.

Example 3.9 suggests the following more delicate strategies for using Theorem 2.10 to obtain a constant-factor approximation algorithm for the MRoB problem.

(1) Modify the AKR-GW algorithm, presumably by forcing it to build a limited number of additional edges, so that there is a local cost-sharing method that is $O(1)$-strict.
(2) Design a non-local $O(1)$-strict cost-sharing method for the AKR-GW algorithm.

As we survey below, both of these approaches have been successfully applied to design $O(1)$-approximation algorithms for the MRoB problem.

3.2.2 *Strict Cost-Sharing Methods for Steiner Forest Algorithms.* The first $O(1)$-strict cost-sharing method for an $O(1)$-approximation algorithm for the Steiner Forest problem was given in the conference version of the present work [Gupta et al. 2003]. Motivated by Example 3.9, this paper shows how to make the AKR-GW algorithm "more aggressive" in a controlled way [Gupta et al. 2003]. This modification forces the algorithm to build additional edges, which in turn simplifies the problem of designing strict cost-shares. More precisely, the main idea in [Gupta et al. 2003] is to run the AKR-GW algorithm for a γ factor longer than usual, where $\gamma \geq 1$ is a parameter. Defining this idea formally requires some care; see [Gupta et al. 2003; Pál 2004] for details. For sufficiently large γ, this algorithm admits an $O(1)$-strict local cost-sharing method [Gupta et al. 2003]. Using Theorem 2.10, these ideas initially gave a 12-approximation algorithm for MRoB [Gupta et al. 2003]; slightly modifying the analysis improved the approximation factor to 8 (see [Pál 2004]).

Next, Becchetti et al. [2005] proposed a different way to force the AKR-GW algorithm to build additional edges. They designed a strict cost-sharing method for their algorithm and obtained an approximation factor of $4 + 2\sqrt{2} \approx 6.83$. This result inspired us to revisit the algorithm of [Gupta et al. 2003] and modify the analysis, resulting in the same approximation ratio of $4 + 2\sqrt{2}$ (see Pál [2004]).

Finally, Fleischer et al. [2006] very recently obtained a better approximation ratio via a different approach. Rather than modifying the AKR-GW algorithm, Fleischer et al. [2006] designed a non-local 3-strict cost-sharing method for it. By Theorem 2.10, this gives a 5-approximation algorithm for MRoB. This is the smallest approximation ratio currently known for the problem. Fleischer et al. [2006] also showed that for every $\beta < 8/3$, there is no β-strict cost-sharing method for the AKR-GW algorithm.

3.3 Multicast Rent-or-Buy

Our techniques also generalize to the MuRoB problem, where there are arbitrary demand groups in place of demand pairs. Formally, an instance of MuRoB is given

by the usual graph $G = (V, E)$ with edge lengths c, a parameter M, and a set $\mathcal{D} = \{D_1, \ldots, D_k\}$ of *demand groups*. Each demand group D_i is an arbitrary set of two or more demands and has a corresponding weight w_i. A feasible solution to a MuRoB instance buys and rents capacity on edges as usual, and also specifies a tree A_i for each demand group D_i that spans all of the demands of D_i. The capacity on each edge e must be at least the weight $\sum_{i\,:\,e\in A_i} w_i$ of the trees that include it. In other words, the capacity installed must be sufficient for simultaneous "multicast" communication within each demand group.

The high-level approach of the Sample-Augment algorithm also applies to the MuRoB problem: sample each demand group D_i independently with probability $\min\{w_i/M, 1\}$, buy infinite capacity on edges to connect demand groups in the randomly sampled subproblem, and greedily rent capacity for the remaining demand groups. The problem that arises in the subproblem step, which we will call the Generalized Steiner Tree (GST) problem [Agrawal et al. 1995; Goemans and Williamson 1995], seems more general than the Steiner Forest problem, since the connectivity requirements now involve demand groups rather than demand pairs. An instance of GST can be converted into an equivalent instance of Steiner Forest, however, for example by replacing each demand group D_i with a set of demand pairs, one for each unordered pair of demands of D_i. Thus every α-approximation algorithm for Steiner Forest can be converted into an α-approximation algorithm for GST. Alternatively, the AKR-GW algorithm can easily be modified to directly approximate the GST problem.

Extending the definition of a strict cost-sharing method is also straightforward. By a *GST cost-sharing method* we mean a function χ that assigns a non-negative cost share $\chi(\mathcal{I}, D_i)$ to each demand group D_i of an instance $\mathcal{I}$ of GST, such that the sum of the cost shares is at most the cost of an optimal solution to $\mathcal{I}$.

Definition 3.10. Let $\mathcal{A}$ be a deterministic algorithm for the GST problem. A GST cost-sharing method χ is *β-strict for $\mathcal{A}$* if for all instances $\mathcal{I} = (G, \mathcal{D})$ of GST and for all demand groups $D_i \in \mathcal{D}$,

$$\ell_{G/F}(D_i) \leq \beta \cdot \chi(\mathcal{I}, D_i),$$

where F is the solution returned for the instance $(G, \mathcal{D} \setminus \{D_i\})$ by the algorithm $\mathcal{A}$, and $\ell_{G/F}(D_i)$ denotes the value of a minimum-cost tree in G/F that spans all of the demands of D_i.

An algorithm for the GST problem is then *β-strict* if it admits some β-strict cost-sharing method. As in Theorem 2.10, if the Sample-Augment algorithm is used with a β-strict α-approximation algorithm for the GST problem, and if the augmentation step is implemented optimally, then it is an $(\alpha+\beta)$-approximation algorithm for MuRoB. One new complication is that computing an optimal solution in the augmentation step of the Sample-Augment algorithm is NP-hard (cf., the proof of Lemma 2.1).

We noted above that an α-approximation algorithm for Steiner Forest naturally induces an α-approximation algorithm for GST. Unfortunately, a strictness guarantee (in the sense of Definition 2.5) for a Steiner Forest approximation algorithm does not necessarily carry over to a strictness guarantee (in the sense of Definition 3.10) for the corresponding GST approximation algorithm. Nonetheless, Gupta and Pál

[2005] proved that the GST algorithm in [Gupta et al. 2003] is $O(1)$-strict, and Fleischer et al. [2006] proved that the AKR-GW GST algorithm is 4-strict. Moreover, the proofs in both of these papers are algorithmic, and can be used to compute a solution in the augmentation step of SAMPLE-AUGMENT in polynomial time. While these solutions are not generally optimal, the guarantee of Theorem 2.10 continues to apply to them [Fleischer et al. 2006; Gupta and Pál 2005]. As a consequence, the MuRoB algorithms in [Gupta and Pál 2005] and [Fleischer et al. 2006] achieve approximation factors of 12.66 and 6, respectively. The latter approximation ratio is the smallest currently known for the problem.

4. VIRTUAL PRIVATE NETWORK DESIGN

In this section and the next, we show that strict cost-sharing methods lead to improved approximation algorithms for two problems to which our analysis framework does not directly apply. In this section, we build on our algorithm and analysis for the SSRoB problem and give a simple 5.55-approximation algorithm for the VPND problem. We study the SSBaB problem in the next section.

4.1 The VPND Algorithm

Recall from Section 1.1 that in an instance of the VPND problem (Problem 1.2) we are given *thresholds* $b_{in}(j)$ and $b_{out}(j)$ on the amount of traffic that enters and leaves each demand $j \in D \subseteq V$ of a network $G = (V, E)$ with edge lengths c_e. The objective is to design a network with sufficient capacity for every traffic pattern that respects these upper bounds. Formally, a traffic pattern is specified by a $D \times D$ matrix of nonnegative real numbers, with entry f_{ij} denoting the amount of traffic sent from demand i to demand j. A traffic matrix is *valid* if for every demand j, the amount of traffic $\sum_i f_{ij}$ incoming to j is at most $b_{in}(j)$ and the amount $\sum_i f_{ji}$ of outgoing traffic is at most $b_{out}(j)$. We assume that all of these thresholds are rational numbers. By scaling both these thresholds and the edge lengths of G, we can then assume, without loss of generality, that these thresholds are integral.

A feasible solution to a VPND instance reserves capacity u_e on each edge e of the graph G, and selects paths P_{ij} between each ordered pair $i, j \in D$ of demands so 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.

To simplify our exposition, we assume for most of this section that each demand j is a either a *sender* (with $b_{in}(j) = 0$ and $b_{out}(j) = 1$) or a *receiver* (with $b_{in}(j) = 1$ and $b_{out}(j) = 0$). In Remark 4.9, we indicate how to extend our algorithm and analysis to general VPND instances. We also assume that the receivers of the VPND instance outnumber the senders; the algorithm and analysis in the other case are symmetric.

Figure 3 presents our algorithm for the VPND problem, which we call VPN-SAMPLE-AUGMENT. Its high-level outline is the same as for the SAMPLE-AUGMENT algorithm. Given an instance $\mathcal{I}$ of VPND, we first define a random subproblem, which in this case is an instance $\mathcal{I}_{SSRoB}$ of SSRoB. The only random parameter of $\mathcal{I}_{SSRoB}$ is the sink vertex, which is a sender $\hat{s}$ of $\mathcal{I}$ that is picked uniformly at random. The source vertices of $\mathcal{I}_{SSRoB}$ are defined to be the receivers of $\mathcal{I}$, and each corresponding demand pair is given unit weight. Finally, the cost M of

Input: an VPND instance (G, D, b).
Assumptions: each demand $j \in D$ is either a sender or a receiver; there are more receivers than senders.

(1) (*Sampling step*) Pick a sender $\hat{s}$ uniformly at random.
(2) (*Subproblem step*) Use the algorithm Sample-Augment to compute a feasible solution to the SSRoB instance $(G, \mathcal{D}, \mathbf{1}, M)$, where $\mathcal{D}$ is the set of all pairs $\{(r, \hat{s}) \mid r \text{ is a receiver}\}$, $\mathbf{1}$ the vector of unit weights, and M is the number of senders in $\mathcal{I}$. Let F denote the edges bought by the algorithm. For every edge $e \in F$, set $u_e = M$; for every other edge e, set u_e equal to the amount of capacity rented for e by the Sample-Augment algorithm.
(3) (*Augmentation step*) Greedily and independently reserve one unit of capacity from each sender other than $\hat{s}$ to F.

Fig. 3. The algorithm VPN-Sample-Augment.

buying capacity on an edge is defined to be the number of senders. We then solve the random subproblem $\mathcal{I}_{SSRoB}$ with the Sample-Augment algorithm of Section 3.1. We interpret the resulting feasible solution of $\mathcal{I}_{SSRoB}$ as follows. Let F be the set of edges on which the Sample-Augment subroutine bought capacity. In our VPND solution, we reserve M units of capacity on each edge $e \in F$. If the Sample-Augment algorithm rents capacity for an edge e, then in our VPND solution we reserve the same amount of capacity on e. Finally, we greedily augment this partial solution to a feasible solution for the VPND instance $\mathcal{I}$ as follows: independently for each sender $s \neq \hat{s}$, reserve one unit of capacity for s's exclusive use on a shortest path between s and F. For each sender s and receiver r, the s-r path P_{sr} is defined as the concatenation of s's shortest path to F, a path through F to $\hat{s}$, and the $\hat{s}$-r path defined by the Sample-Augment subroutine's solution to the instance $\mathcal{I}_{SSRoB}$.

To gain some intuition behind the VPN-Sample-Augment algorithm, consider the case when there is only one sender in the system: the optimal solution in that case is a minimum-cost Steiner tree on this sender and all the receivers. And indeed, the algorithm VPN-Sample-Augment will set $M = 1$ and hence sample all receivers, and will build a low-cost Steiner tree on them. The other extreme is when there is an equal number M of senders and receivers. In this case, consider the "uniform" traffic pattern with $1/M$ amount of traffic between each sender-receiver pair—any solution to the VPND problem must route this traffic pattern. Moreover, one solution is to pick a random sender σ and receiver ρ: all senders send their traffic to the random receiver ρ, who forwards these M units of traffic to the random sender σ, who in turn forwards the traffic to the final destinations. A little thought shows that the cost of each of these three forwarding steps, averaged over the random choices of σ and ρ, is at most the cost of routing the uniform traffic pattern. This gives a 3-approximation algorithm. Moreover, the VPN-Sample-Augment algorithm performs similarly on this instance, picking one random sender and (in expectation) one random receiver. Finally, on general instances, the algorithm VPN-Sample-Augment tries to combine its actions on these two extreme cases, which is also reflected in our analysis below.

To begin the analysis, we prove some basic facts about the algorithm VPN-Sample-Augment. For the remainder of the analysis, fix an instance $\mathcal{I} = (G, D, b)$

of VPND that satisfies our two standing assumptions. Let R and S denote the sets of receivers and senders of $\mathcal{I}$, respectively. Let F denote the set of edges bought by the Sample-Augment algorithm in the subproblem step of VPN-Sample-Augment.

Lemma 4.1. *The algorithm* VPN-Sample-Augment *produces a feasible solution with probability 1.*

Proof. Fix a valid demand matrix $\{f_{sr}\}_{s\in S, r\in R}$. We need to show that routing f_{sr} units of flow on the path P_{sr} defined above for every $s \in S$ and $r \in R$ does not violate any capacity constraint (with probability 1). We first claim that no edge $e \in F$ bought by the Sample-Augment subroutine in the subproblem step is used beyond its capacity. This follows because M units of capacity are reserved on each such edge and, since there are only M senders, $\sum_{s,r} f_{sr} \leq \sum_s b_{out}(s) = M$.

On the other hand, the VPN-Sample-Augment algorithm explicitly reserves capacity on each edge outside F for each path that uses it. In more detail, for every sender s, all paths of the form P_{sr} begin with a shortest path from s to F, and the augmentation step of the VPN-Sample-Augment algorithm reserves one unit of capacity on this subpath for exclusive use by s. Since $\sum_r f_{sr} \leq b_{out}(s) = 1$, there is sufficient capacity for the traffic on these subpaths. Similarly, for each receiver r, all paths of the form P_{sr} conclude with the $\hat{s}$-r path P_r defined by the Sample-Augment algorithm's solution to the instance $\mathcal{I}_{SSRoB}$. Moreover, $\sum_s f_{sr} \leq 1$. By the definition of the augmentation step of the Sample-Augment algorithm, there is one unit of capacity on the edges of $P_r \setminus F$ reserved for exclusive use by the sender r. There is thus sufficient capacity on every edge for the traffic of every path P_{sr}, and the proof is complete. □

Also, the union of the routing paths produced by the VPN-Sample-Augment algorithm form a tree with probability 1.

Lemma 4.2. *If a consistent tie-breaking rule is used to compute shortest paths, then with probability 1 the algorithm* VPN-Sample-Augment *produces a solution in which the edges with non-zero capacity form a tree.*

Proof. Since the set F is the output of a Steiner Tree instance algorithm, it is (or can be assumed to be) a tree. By the definition of the augmentation steps of the Sample-Augment and VPN-Sample-Augment algorithms, all other edges with non-zero capacity lie on a shortest path between a demand j and the set F—equivalently, are contained in the shortest-path tree in the contracted graph G/F rooted at the vertex corresponding to F. This implies that if a consistent tie-breaking rule is used to compute shortest paths, the set of all edges with non-zero capacity forms a tree. □

4.2 Analysis

We now bound the expected cost of the solution produced by the VPN-Sample-Augment algorithm for the VPND instance $\mathcal{I}$. We bound three parts of this cost separately: the expected cost corresponding to the set F of edges bought by the Sample-Augment subroutine in the subproblem step; the expected cost corresponding to the rented edges in the subproblem step; and the expected cost of the augmentation step. The first two steps hinge on the following lemma, which

bounds the expected cost of an optimal solution to the (random) instance of Steiner Tree that arises in the subproblem step of the Sample-Augment subroutine (cf., Lemma 2.2).

Lemma 4.3. *Let OPT_{VPN} denote the cost of an optimal solution for the* VPND *instance $\mathcal{I}$. Let $OPT_{\hat{s},\hat{R}}$ denote the cost of an optimal solution for the* Steiner Tree *instance in the subproblem step of the* Sample-Augment *subroutine, given the random choices of the sender $\hat{s} \in S$ and receivers $\hat{R} \subseteq R$ in the sampling steps of the* VPN-Sample-Augment *and* Sample-Augment *algorithms, respectively. Then*

$$\mathbf{E}\left[OPT_{\hat{s},\hat{R}}\right] \leq \frac{OPT_{VPN}}{M}, \tag{9}$$

where the expectation is over the random choices of $\hat{s}$ and $\hat{R}$.

Proof. We begin with the following equivalent description of the random choices made in the sampling steps of the VPN-Sample-Augment and Sample-Augment algorithms. Suppose each receiver picks a sender independently and uniformly at random. Let $D_s \subseteq R$ denote the random set of receivers that pick the sender s. Then, independently choose a sender $\hat{s}$ uniformly at random and consider the Steiner Tree instance $\mathcal{I}_{\hat{s}}$ defined by $D_{\hat{s}} \cup \{\hat{s}\}$. We claim that this random process induces the same distribution over Steiner Tree instances that the algorithm VPN-Sample-Augment does. In both processes, one sender $\hat{s}$, chosen uniformly at random from the set of all senders, is included in the Steiner Tree instance. In the VPN-Sample-Augment algorithm, each receiver has a $1/M$ probability of being included in the Steiner Tree instance by the definition of the sampling step of the Sample-Augment subroutine. In the new random process, since there are M senders, the probability that a receiver picks the sender $\hat{s}$ and is included in the resulting Steiner Tree instance is also $1/M$. Moreover, these events are independent of each other and of the choice of the sender $\hat{s}$, just as in the VPN-Sample-Augment algorithm. The two random processes therefore induce the same distribution over Steiner Tree instances, and we can prove the lemma by establishing (9) for the new random process above.

We now prove that the expected cost of an optimal solution to the random Steiner Tree instance $\mathcal{I}_{\hat{s}}$ is at most OPT_{VPN}/M. We prove this inequality after conditioning on the partition $\{D_s\}_{s\in S}$ of receivers, with the expectation only over the choice of $\hat{s}$; the unconditional inequality (9) then follows. Fix an optimal solution to the VPND instance $\mathcal{I}$ that reserves the paths $\{P^*_{sr}\}_{s\in S, r\in R}$ and capacities $\{u^*_e\}_{e\in E}$. We next show how to pack feasible solutions for all M of the Steiner Tree instances $\{\mathcal{I}_s\}_{s\in S}$ into this optimal solution.

For each sender $s \in S$, let G^*_s denote the subgraph of G with the edge set $\cup_{r\in D_s} P^*_{sr}$. Since G^*_s spans $D_s \cup \{s\}$, the cost $c(G^*_s)$ of the subgraph G^*_s is at least denote the cost OPT_s of an optimal solution to $\mathcal{I}_s$. Moreover, if an edge e appears in k 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_{VPN} must install at least k units of capacity on the edge e.

Therefore,

$$OPT_{VPN} \geq \sum_{s \in S} c(G_s^*) \geq \sum_{s \in S} OPT_s.$$

Thus, if we pick a sender uniformly at random from the M senders, $\mathbf{E}_s[OPT_s] \leq OPT_{VPN}/M$, which completes the proof. □

A proof identical to that of Lemma 2.3 bounds the expected cost incurred by the VPN-Sample-Augment algorithm for bought edges in its subproblem step.

Lemma 4.4. *If an α-approximation algorithm for* Steiner Tree *is used in the subproblem step of the* Sample-Augment *subroutine, then the expected cost incurred by the* VPN-Sample-Augment *algorithm for bought edges in its subproblem step is at most $\alpha \cdot OPT_{VPN}$.*

We next use the universally strict cost shares for Steiner Tree (Section 3.1) to bound the expected cost incurred by the VPN-Sample-Augment algorithm in the subproblem step for edges that were rented by its Sample-Augment subroutine.

Lemma 4.5. *The expected cost incurred by the* VPN-Sample-Augment *algorithm for rented edges in its subproblem step is at most $2 \cdot OPT_{VPN}$.*

Proof. Let C denote the cost paid by the VPN-Sample-Augment algorithm for rented edges in its subproblem step. Recall from Definition 3.2 and Example 3.3 that the Prim cost-sharing method of Example 2.8 is universally 2-strict. In particular, Lemma 3.4 implies that these cost shares are 2-strict no matter what Steiner Tree algorithm is used in the subproblem step of the Sample-Augment algorithm.

We next condition on the choice of $\hat{s}$ in the sampling step of the VPN-Sample-Augment algorithm. For a subset $\hat{R} \subseteq R$ of receivers, let $OPT_{\hat{s},\hat{R}}$ denote the value of a minimum-cost Steiner tree spanning $\hat{s}$ and all of the receivers in $\hat{R}$. The proof of Lemma 2.9, and the inequalities (4) and (8) in particular, imply that

$$\mathbf{E}_{\hat{R}}[C|\hat{s}] \leq 2M \cdot \mathbf{E}_{\hat{R}}\left[OPT_{\hat{s},\hat{R}}|\hat{s}\right],$$

where the expectations are over the random choice of the set $\hat{R}$ of receivers in the Sample-Augment subroutine's sampling step. Taking expectations over the choice of $\hat{s}$, we obtain

$$\mathbf{E}_{\hat{s},\hat{R}}[C] \leq 2M \cdot \mathbf{E}_{\hat{s},\hat{R}}\left[OPT_{\hat{s},\hat{R}}\right] \leq 2 \cdot OPT_{VPN},$$

where the second inequality follows from Lemma 4.3. The proof is complete. □

Our final lemma bounds the expected cost of the augmentation step of the VPN-Sample-Augment algorithm.

Lemma 4.6. *The expected cost incurred in the augmentation step of the* VPN-Sample-Augment *algorithm is at most $2 \cdot OPT_{VPN}$.*

Proof. Since the set F of bought edges contains the sender $\hat{s}$, we can prove the lemma by showing that, if a sender $\hat{s}$ is picked uniformly at random, then

$$\mathbf{E}\left[\sum_{s \in S} \ell(s, \hat{s})\right] \leq 2 \cdot OPT_{VPN},$$

where $\ell(\cdot,\cdot)$ denotes shortest-path distance in G. To prove this inequality, we fix an arbitrary set $\hat{R} \subseteq R$ of M receivers. Every perfect matching $\mathcal{M}$ of S and $\hat{R}$ provides a lower bound $\sum_{(s,r)\in\mathcal{M}} \ell(s,r)$ on OPT_{VPN}, since a feasible solution must support the simultaneous communication of all of the matched pairs of $\mathcal{M}$. Averaging over all of the $M!$ possible perfect matchings of S and $\hat{R}$, we obtain

$$\frac{1}{M} \sum_{s\in S, r\in\hat{R}} \ell(s,r) \leq OPT_{VPN},$$

as each sender-receiver pair (s,r) appears in $(M-1)!$ of the $M!$ perfect matchings. This inequality implies that

$$\mathbf{E}_{\hat{s}} \left[\sum_{r\in\hat{R}} \ell(\hat{s},r) \right] \leq OPT_{VPN}. \tag{10}$$

Also, by the Triangle inequality for shortest-path distances,

$$\sum_{s\in S} \ell(s,\hat{s}) \leq \sum_{r\in\hat{R}} \ell(\hat{s},r) + \sum_{(s,r)\in\mathcal{M}} \ell(s,r) \leq \sum_{r\in\hat{R}} \ell(\hat{s},r) + OPT_{VPN}, \tag{11}$$

where $\mathcal{M}$ is an arbitrary perfect matching of S and $\hat{R}$. Taking expectations (over the choice of $\hat{s}$) in (11) and combining with (10) proves the lemma. □

Combining Lemmas 4.4–4.6 with the 1.55-approximation algorithm for the Steiner Tree problem due to Robins and Zelikovsky [2005] yields the main theorem of this section.

THEOREM 4.7. *There is a randomized 5.55-approximation algorithm for the* VPND *problem.*

Lemma 4.2 states that the VPN-SAMPLE-AUGMENT algorithm always outputs a tree solution. Our analysis of the algorithm, however, does not assume that the paths chosen by the optimal solution form a tree. Indeed, there are instances in which no optimal solution forms a tree [Gupta et al. 2001]. Theorem 4.7 implies that for every instance of VPND, there is a tree solution within a small constant factor of the optimal (graph) solution. This resolves one of the main open questions from [Gupta et al. 2001].

COROLLARY 4.8. *Every instance of* VPND *admits a tree solution with cost no more than 5.55 times that of an optimal (graph) solution. Moreover, this solution can be computed in polynomial time.*

If the constraint of polynomial-time computation is dropped, then the constant in Corollary 4.8 can be improved to 5 by using an (exponential-time) optimal Steiner Tree subroutine in the VPN-SAMPLE-AUGMENT algorithm.

Remark 4.9. The VPN-SAMPLE-AUGMENT algorithm and its analysis extend to the case of arbitrary (integral) thresholds b_{in} and b_{out} as follows. Given an instance of VPND, suppose we modify the instance by splitting each demand j into $b_{in}(j)$ receivers and $b_{out}(j)$ senders, all of which are co-located. This increases the set of feasible solutions, since it allows the traffic of an original demand pair to be routed

on more than one path. The modification can therefore only decrease the cost of an optimal solution. On the other hand, if the VPN-Sample-Augment algorithm uses a consistent tie-breaking rule for computing shortest paths as in Lemma 4.2, then it will output a solution for the modified instance that is also feasible for the original instance. Running the VPN-Sample-Augment algorithm after splitting demands into senders and receivers therefore produces a feasible solution to the original instance that has cost at most 5.55 times an optimal solution (for the original or the modified instance).

Splitting demands into senders and receivers is only a polynomial transformation if all of the demand thresholds are polynomially bounded. However, by adjusting the sampling probabilities in the sampling steps of the VPN-Sample-Augment algorithm and its Sample-Augment subroutine, we can easily modify the algorithm to mimic its behavior on the modified instance in polynomial time.

5. SINGLE-SINK BUY-AT-BULK NETWORK DESIGN

This section gives a simple constant-factor approximation algorithm for the widely studied SSBaB problem. Our algorithm is closely related to that of Guha et al. [2001], but the analysis tools developed in this paper permit a tighter and equally simple analysis. Section 5.1 introduces notation for our analysis and reviews some well-known transformations of SSBaB instances. Section 5.2 presents our algorithm and analysis.

5.1 Preliminaries

Recall that an instance of the SSBaB problem (Problem 1.3) comprises an undirected graph G and edge costs c; a set $\mathcal{D}$ of demand pairs $\{(s_i, t)\}_{i=1}^{k}$; a weight $w_i \geq 0$ for each demand pair (s_i, t), denoting the amount of flow that s_i wants to send to t; and K *cable types* $\{1, 2, \ldots, K\}$, where the jth cable has capacity u_j and cost σ_j per cable per unit length. The goal is to compute a minimum-cost way of installing cables so that there is sufficient capacity for all sources to route flow simultaneously.

Fix an instance $\mathcal{I}$ of SSBaB. We will assume that each parameter u_j and σ_j is a power of 2. Similarly to [Guha et al. 2001], this assumption can be enforced while losing a factor of 4 in the approximation ratio, by rounding each capacity u_j down to the nearest power of 2 and each σ_j up to the nearest power of 2. By scaling and reordering cable types, we can assume that $1 = u_1 < \cdots < u_K$ and $1 = \sigma_1 < \cdots < \sigma_K$; if $u_i \leq u_j$ and $\sigma_i \geq \sigma_j$, then cable type i is redundant and can be eliminated (since one can replace each cable of type i with a cable of type j maintaining feasibility without increasing the cost).

Define $\delta_j = \sigma_j / u_j$, which intuitively is the "incremental cost" of using cable type j. For all j, δ_j is a power of 2. We can assume that $\delta_1 > \ldots > \delta_K$, since if $\delta_i \leq \delta_j$ for some $i < j$, then cable type j is redundant and can be eliminated: again one can replace each cable of type j with a cable of type i to get a feasible solution with no higher cost.

Finally, we define $g_j = \frac{\sigma_{j+1}}{\sigma_j} u_j$ and $g_K = \infty$; in other words, g_j is the amount of flow for which the cost of routing using cable types j and $j+1$ is the same (and hence for values of flow higher than g_j, it makes sense to use cable type $j+1$).

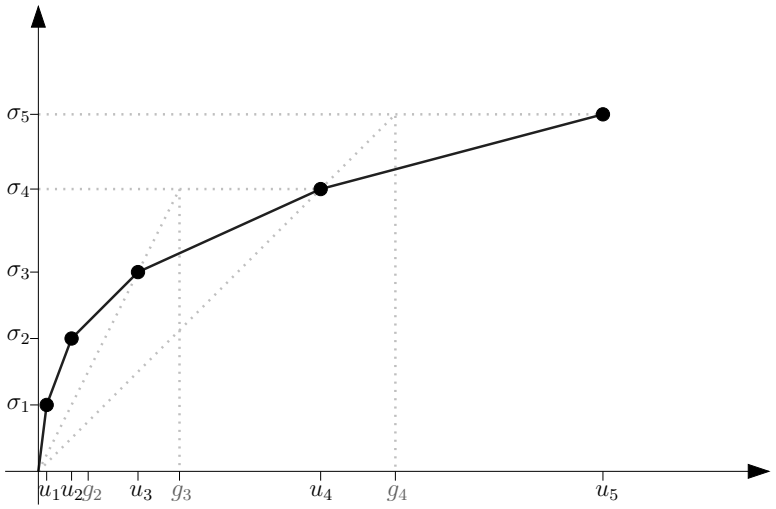

Fig. 4. The cable-installation cost function.

Since $\delta_j > \delta_{j+1}$, $g_j < u_{j+1}$ and hence

$$1 = u_1 < g_1 < u_2 < g_2 < \ldots < u_K < g_K = \infty. \tag{12}$$

See also Figure 4.

Next, we would like to assume that all weights w_i are integral. This assumption is not without loss of generality, as we have already scaled the cable capacities. Instead, we enforce this assumption with the following "redistribution lemma," which allows us to take the non-integral weights at nodes and move them around in order to collect an integral amount of weight at some subset of the nodes. Roughly speaking, this lemma shows how to take a grouping parameter U (which would be 1 to ensure integral demands) along with a tree with weights on its vertices, and randomly define a flow that moves weights throughout the tree so that the total weight at every node of the tree becomes either 0 or U. (One should think of the weight as specifying the amount of "material" at each node, and the flow as specifying how to move this material.) 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 5.1 **Redistribution Lemma**. *Let T be a tree and $U > 0$ a parameter. Suppose each vertex $j \in T$ has a nonnegative weight $w_j < U$ and that the sum $\sum_j w_j$ of the weights is a multiple of U. Then there is an efficiently computable (random) flow f in T with the following properties.*

(a) With probability 1, f sends at most U units of flow across each edge of T.

(b) After rerouting weights according to the flow f, for every vertex $j \in T$, the new weight of j is U with probability w_j/U and 0 with probability $1 - w_j/U$.

A deterministic version of this lemma appears in [Hassin et al. 2004, Lemma 1]. We include the simple proof for completeness.

PROOF. Replace each edge of T by two oppositely directed arcs. We first show that the lemma holds in this bidirected tree $\widetilde{T}$. We start by rooting $\widetilde{T}$ at an arbitrary vertex r and taking an Euler tour of $\widetilde{T}$ starting at r. Order the vertices $j_1, \ldots, j_n$ of T according to their first appearance in this Euler tour. For each $i \in \{1, 2, \ldots, n\}$, let W_i denote the sum of the weights of the first i vertices in this ordering. Define W_0 to be 0.

Pick a value Y drawn uniformly at random from $(0, U]$. Call vertex j_i *unlucky* if for some integer x, $W_{i-1} < xU+Y \leq W_i$—if the running sum of weights just crossed the point Y modulo U—and *lucky* otherwise. After this procedure concludes, we define the flow $\widetilde{f}$ to reroute weights as follows. If a vertex j_i is lucky, we add a flow path to $\widetilde{f}$ that routes all of j_i's weight to the unlucky vertex that is next according to the ordering $j_{i+1}, \ldots, j_n, j_1, \ldots, j_{i-1}$. Otherwise, the vertex j_i is only allowed to route $W_i - (xU + Y)$ units of its weight to the next unlucky vertex, where x is the integer defined above.

After this rerouting, a vertex has weight U if it is unlucky and weight 0 if it is lucky. The probability that the vertex j is unlucky is precisely w_j/U. Thus the flow $\widetilde{f}$ satisfies part (a) of the lemma. The flow need not satisfy part (b), however: while $\widetilde{f}$ routes at most U units of flow on each arc of $\widetilde{T}$, this corresponds to routing at most $2U$ units of flow on each edge of the original undirected tree T. But since $\widetilde{f}$ routes at most U units of flow in each direction across each edge of T, we can perform rudimentary flow-canceling independently on each edge of T. This yields a flow f in T that satisfies part (b) of the lemma and, since it redistributes weights identically to $\widetilde{f}$, also satisfies part (a). □

We will use Lemma 5.1 as a preprocessing step to collect integral demands at some subset of the sources of the instance $\mathcal{I}$. First, we can assume that the sum of the demand pair weights in $\mathcal{I}$ is greater than 1; otherwise even the cheapest cable type effectively has infinite capacity, and $\mathcal{I}$ is equivalent to a Steiner Tree instance. We also assume that the sum W of the demand pair weights in $\mathcal{I}$ is a power of 2 and is at least u_K; this assumption can be removed by adding a dummy demand pair (t, t) with an appropriate weight, and getting an approximation for this new instance. (This follows from the fact that an optimal solution to the original instance is also an optimum for the new instance with the dummy pair; moreover, any solution to the new instance induces a solution to the original instance with no greater cost, since the new instance has only more flow to route.)

As a preprocessing step of the algorithm in the next subsection, we use an α-approximation algorithm for the Steiner Tree problem to compute a tree T_0 that spans all of the sources, and build one cable of type 1 on each edge of T_0. We then apply Lemma 5.1 to the tree T_0, with $U = 1$ and the weight of the source s_i defined as the fractional part $w_i - \lfloor w_i \rfloor$ of its weight in $\mathcal{I}$. After this procedure concludes, there is an integral amount of weight at every source of $\mathcal{I}$.

We now bound the cost of T_0. Fix an optimal solution to $\mathcal{I}$ and let OPT denote its cost. Let $C^*(j)$ denote the cost of the cables of type j in this solution. Note that $OPT = \sum_{j=1}^{K} C^*(j)$. This solution must install nonzero capacity on a subgraph

G^* of G that spans all of the sources of $\mathcal{I}$. Thus one candidate for a Steiner Tree solution T_0 is to build one type 1 cable on each edge of G^*. Since $\sigma_1 = 1$, the cost of this candidate solution is at most

$$\sum_{j=1}^{K} \frac{C^*(j)}{\sigma_j}. \tag{13}$$

Since we use an α-approximation algorithm to compute the Steiner tree solution T_0, the cost of T_0 is at most α times the quantity in (13).

5.2 The Algorithm SSBaB-Sample-Augment

We now present our constant-factor approximation algorithm for the SSBaB problem. The algorithm is similar to that of Guha et al. [2001], where the network is designed incrementally in stages. At the beginning of each stage j there will be a set of *demands*, each of which represents a group of u_j units of traffic that must be routed to the sink. During the jth stage, we use the value u_{j+1} as an "aggregation threshold", and reroute groups of u_{j+1}/u_j demands (each of weight u_j) into a single demand of weight u_{j+1}. We buy cables on the paths required for this agglomeration. At the end of all of the stages, every demand reaches the sink. The final solution is the union of all of the cables bought in all of the stages. Since this capacity is sufficient to move all of the prescribed traffic from the sources to the sink (via the concatenation of the rerouting paths used in each stage of the algorithm), this solution is feasible.

Let W denote the sum of the demand pair weights; recall from Section 5.1 that we can assume that W is a power of 2. Our preprocessing step from Section 5.1 ensures that at the beginning of the first stage there is an integral weight at every source vertex. If the source s_i has weight w_i at the beginning of the first stage, we interpret this as w_i co-located demands, each of weight 1. Let D_1 denote the set of these unit-weight demands. While naively replicating demands could result in a pseudopolynomial-time algorithm, non-uniform sampling can be added to the SSBaB-Sample-Augment algorithm to simulate the effect of this replication in polynomial time (see also Remark 4.9).

More generally, at the beginning of the jth stage, there is a set D_j of W/u_j demands, located at the source vertices of $\mathcal{I}$, with weight u_j each. We now describe each stage j of the algorithm in more detail; see also Figure 5. In the sampling step, we choose a random subset $\widehat{D}_j \subseteq D_j$ of demands, with each demand of D_j picked independently with probability $p_j = u_j/g_j = \sigma_j/\sigma_{j+1}$. Note that the sampling probability p_j is the ratio between the costs of the relatively low-capacity type j cables and relatively high-capacity type $(j+1)$ cables, analogous to the sampling step in the algorithm Sample-Augment for rent-or-buy problems. In the subproblem step, we compute a Steiner tree T_j spanning the set F_j, which is the union of the sink t and the source vertices that contain a demand of $\widehat{D}_j$. We build one cable of type $(j+1)$ on each edge of T_j. In the augmentation step, we route the demands outside $\widehat{D}_j$ to vertices of F_j along shortest paths, while building cables of type j on these shortest paths. In the gathering step, for each co-located group of u_{j+1}/u_j demands, we send all of these demands back to the originating location (at the beginning of this stage) of one of them, chosen uniformly at random. This

(1) (*Sampling step*) Construct a random subset $\widehat{D}_j$ of the demands in D_j by choosing each such demand independently with probability $p_j = u_j/g_j = \sigma_j/\sigma_{j+1}$.
(2) (*Subproblem step*) Let F_j denote the union of the sink and the sources that contain a demand from $\widehat{D}_j$. Construct an α-approximate Steiner tree T_j that spans F_j. Install a cable of type $(j+1)$ on each edge of T_j.
(3) (*Augmentation step*) For each demand in D_j, route its u_j weight to the closest vertex in F_j. (Since $\widehat{D}_j \subseteq F_j$, none of the weight at vertices in $\widehat{D}_j$ needs to be routed.) Install one cable of type j on each edge of this shortest path.
(4) *(Gathering step)* For each vertex $v \in F_j$, split the demands at v into *complete groups* of u_{j+1}/u_j demands plus one *residual group* of $r_v < u_{j+1}/u_j$ demands. Route each complete group back to the initial location (at the beginning of this stage) of one of the u_{j+1}/u_j contributing demands (using shortest paths), chosen uniformly at random. Install cables of type $j+1$ to provide sufficient capacity.
(5) *(Rounding step)* Use Lemma 5.1 with the tree T_j, the parameter $U = u_{j+1}$, and the weights of the residual groups, to aggregate the weight of all of the residual groups into complete groups of u_{j+1}/u_j demands, each with total weight exactly u_{j+1}. Reroute a complete group at the vertex $v \in F_j$ back to the initial location of one of the r_v demands that were routed to v in the augmentation step, chosen uniformly at random. Again, build new cables of type $j+1$ to provide sufficient capacity.

Fig. 5. The jth stage of the algorithm SSBaB-Sample-Augment.

group of u_{j+1}/u_j demands is then treated as a single demand of D_{j+1} with weight u_{j+1} in the next stage. Finally, the rounding step is like the preprocessing step of Section 5.1 and uses Lemma 5.1 to gather the remaining demands into groups of u_{j+1}/u_j demands. Each such group is then rerouted using a process similar to that in the gathering step (see Figure 5 for a precise definition), and forms a single demand of D_{j+1} of weight u_{j+1} in the next stage. In the Kth stage, $g_K = \infty$ and $p_K = 0$. Thus, the sampling step of the final stage is vacuous and all demands are sent to the sink t in the augmentation step.

Each demand d of D_{j+1} can be naturally associated with a demand of D_j—the demand that participated in the complete group of u_{j+1}/u_j demands of D_j that corresponds to d, and that was randomly chosen in the gathering or rounding step. Put differently, we can view the jth stage of the algorithm as, for each complete group of u_{j+1}/u_j demands identified in the gathering and rounding steps, multiplying the weight of a random such demand by a u_{j+1}/u_j factor and discarding the rest of them. We can thus sensibly write $D_{j+1} \subseteq D_j$ for every $j \in \{1, 2, \ldots, K-1\}$. Finally, recall that D_1 is the result of the preprocessing step of Section 5.1 and is not the original set of demands of $\mathcal{I}$. Define D_0 as the initial set of demands, with each demand pair (s_i, t) with weight w_i of $\mathcal{I}$ giving rise to $\lceil w_i \rceil$ demands of D_0 ($\lfloor w_i \rfloor$ unit-weight demands and one demand with weight $w_i - \lfloor w_i \rfloor$). Lemma 5.1(b) implies that the probability that a demand of D_0 is also in D_1 is exactly its weight.

We now analyze the algorithm on the fixed SSBaB instance $\mathcal{I}$ with a sequence of lemmas.

Lemma 5.2. *For every unit-weight demand $d \in D_1$ and every stage $j \in \{1, 2, \ldots, K\}$,*

$$\mathbf{Pr}[d \in D_j] = \frac{1}{u_j}.$$

PROOF. The proof is by induction. The lemma is clearly true when $j = 1$ since $u_1 = 1$. For $j > 1$, we have

$$\mathbf{Pr}[d \in D_j] = \mathbf{Pr}[d \in D_j \,|\, d \in D_{j-1}] \cdot \mathbf{Pr}[d \in D_{j-1}].$$

Since $\mathbf{Pr}[d \in D_{j-1}] = 1/u_{j-1}$ by the inductive hypothesis, we only need to show that $\mathbf{Pr}[d \in D_j \,|\, d \in D_{j-1}] = u_{j-1}/u_j$. If d is gathered into a complete group of u_j/u_{j-1} demands in the gathering step of stage $(j-1)$ of the algorithm, then this equality holds because every such demand is equally likely be chosen for membership in D_j. Suppose d is gathered into a residual group of $r_v < u_j/u_{j-1}$ demands at the vertex $v \in F_{j-1}$ in the gathering step of stage $(j-1)$ of the algorithm. Then d is included in D_j if and only if the Redistribution Lemma gathers a complete group of demands at the vertex v in the rounding step and then d is chosen for membership in D_j from the r_v demands in the residual group at v. By Lemma 5.1(b), the probability of both events occurring is

$$\frac{r_v u_{j-1}}{u_j} \cdot \frac{1}{r_v} = \frac{u_{j-1}}{u_j},$$

which completes the proof of the lemma. □

Lemma 5.2 implies that for every stage $j \in \{1, 2, \ldots, K\}$, a demand $d \in D_1$ lies in the set $\widehat{D}_j$ with probability $p_j \times 1/u_j = 1/g_j$. The probability that a demand $d \in D_0$ with weight $w \leq 1$ lies in the set $\widehat{D}_j$ is thus w/g_j.

The next lemma bounds the expected cost of an optimal solution to the Steiner Tree instance arising in the subproblem step of each stage of the SSBAB-SAMPLE-AUGMENT algorithm (cf., Lemmas 2.2 and 4.3).

LEMMA 5.3. *For a stage $j \in \{1, 2, \ldots, K-1\}$, let T_j^* be a minimum-cost Steiner tree spanning F_j with cost $c(T_j^*)$. Then*

$$\mathbf{E}\left[c(T_j^*)\right] \;\leq\; \sum_{i=j+1}^{K} \frac{C^*(i)}{\sigma_i} + \frac{1}{g_j} \sum_{i=1}^{j} \frac{C^*(i)}{\delta_i}, \tag{14}$$

where $C^*(i)$ denotes the cost of the cables of type i in a fixed optimal solution to $\mathcal{I}$, and the expectation is over the choice of $\widehat{D}_j$.

PROOF. As in the proof of Lemma 2.2, we will exhibit a (random) subgraph G_j of G that spans F_j and has low expected cost. Fix an optimal solution for $\mathcal{I}$ and a feasible way of routing all of the traffic with respect to this solution. We first add to G_j all of the edges in the optimal solution that possess a cable of type $j+1$ or higher. The cost of these edges is (deterministically) at most the first sum on the right-hand side of (14).

We complete G_j by considering each demand d of $\widehat{D}_j$ in turn. In the fixed optimal solution, the traffic of the corresponding demand $d \in D_0$ may be routed on multiple paths. (We unfortunately cannot assume without loss of generality that an optimal solution is a tree.) We randomly add to G_j one of these paths, with a path chosen with probability equal to the fraction of d's traffic that it carries.

We now bound the expected cost of adding these edges to G_j. Consider an edge e of G with no cable of type $j+1$ or higher in the optimal solution. First suppose

that only one cable is installed on e, say of type $i \leq j$. Then e is included in the random subgraph G_j if and only if the following events occur: for some demand $d \in D_0$ and some path P that routes some of d's traffic across the edge e, the demand d lies in $\widehat{D}_j$, and the path P is selected among all paths that route d's traffic. A demand $d \in D_0$ with weight $w \leq 1$ lies in $\widehat{D}_j$ with probability w/g_j, and a path P is chosen with probability x/w, where x is the amount of d's traffic that is routed on P in the optimal solution. The Union Bound then implies that e lies in G_j with probability at most f_e/g_j, where f_e is the total amount of flow on e in the optimal solution.

Since $f_e \leq u_i$, edge e contributes at most $c_e u_i/g_j$ to the expected cost of G_j. On the other hand, the cable of type i on edge e contributes $\sigma_i c_e$ to $C^*(i)$. Thus the expected cost in G_j for edge e is at most $1/(g_j\,\delta_i)$ times what the optimal solution pays for the cable. For edges on which the optimal solution installs multiple cables, a similar analysis has to be performed on a cable-by-cable basis. Indeed, in this case, we can emulate multiple cables on a single edge via several parallel copies of the same edge with a single cable on each, and apply the same analysis as above. (Recall that we do not assume that the optimal solution is a tree, and hence making these parallel copies does not affect the argument.) Summing over all edges with no cable of type $j+1$ or higher in the optimal solution gives the second sum on the right-hand side of (14) and proves the lemma. □

We now relate the expected cost incurred by the SSBaB-Sample-Augment algorithm to the expected cost of an optimal Steiner tree spanning the vertices in F_j.

Lemma 5.4. *Let $j \in \{1, 2 \ldots, K-1\}$ be a stage and T_j^* a minimum-cost Steiner tree spanning F_j with cost $c(T_j^*)$. The expected cost incurred in stage j of the algorithm* SSBaB-Sample-Augment *is at most*

$$(3+\alpha)\,\sigma_{j+1}\,\mathbf{E}\,[c(T_j^*)],$$

where α is the approximation ratio of the Steiner Tree *algorithm used in the subproblem step.*

Proof. Since we install one cable of type $(j+1)$ on each edge of the tree T_j that we compute in the subproblem step, the expected cost incurred in this step is at most $\alpha\,\sigma_{j+1}\,\mathbf{E}\,[c(T_j^*)]$. As in Lemma 4.5, the universally 2-strict Prim cost shares of Example 2.8 imply that the expected cost of the augmentation step is at most $2\,\sigma_{j+1}\,\mathbf{E}\,[c(T_j^*)]$. In more detail, we abuse notation and write $\chi(\widehat{D}_j, d_i)$ for the Prim cost share of a demand pair (d_i, t) in the Steiner Tree instance $(G, \mathcal{D})$, where $\mathcal{D} = \{(d_i, t) : d_i \in \widehat{D}_j\}$. As in the proof of Lemma 2.9, we define two random variables B_i and R_i for each demand $d_i \in D_j$. The random variable B_i is equal to σ_{j+1} times the Prim cost share $\chi(\widehat{D}_j, d_i)$ when $d_i \in \widehat{D}_j$, and to 0 otherwise. Note that by Definition 2.4, $\sum_{d_i \in D_j} B_i$ is at most $\sigma_{j+1}\,c(T_j^*)$ (with probability 1).

The variable R_i is defined to be zero when $d_i \in \widehat{D}_j$, and is equal to σ_j times the length $\ell(d_i, F_j)$ of a shortest path between d_i and a vertex of F_j. Since the probability that d_i lies in $\widehat{D}_j$ is $p_j = \sigma_j/\sigma_{j+1}$, we can follow the proof of Lemma 2.9 and infer an upper bound on the expected cost of the augmentation step thus: $\mathbf{E}\,[\sum_i R_i] \leq 2 \cdot \mathbf{E}\,[\sum_i B_i] \leq 2\sigma_{j+1} \cdot \mathbf{E}\,[c(T_j^*)]$.

We complete the proof by showing that the expected cost of the gathering and rounding steps is at most $\sigma_{j+1}\,\mathbf{E}\,[c(T_j^*)]$. Intuitively, we will charge the expected cost of these steps to that of the earlier augmentation step. Lemma 5.1 ensures that the rerouting of residual demands in the rounding step can be accomplished using the cables of type $j+1$ purchased in the subproblem step, and no new cables need to be built. For every edge e of G, one cable of type j was installed on e in the augmentation step of the jth stage for each demand of D_j that used e to travel to a vertex of F_j. In the gathering and rounding steps, one cable of type $(j+1)$ is installed on e for each such demand that is chosen for membership in the set D_{j+1}. Recall from the proof of Lemma 5.2 that for every demand $d \in D_j$, the probability that d is included in D_{j+1} is precisely u_j/u_{j+1}. The expected cost of rerouting demands in the gathering and rounding steps is therefore at most

$$\frac{u_j}{u_{j+1}} \cdot \frac{\sigma_{j+1}}{\sigma_j} = \frac{\delta_{j+1}}{\delta_j}$$

times the expected cost of the augmentation step. Since $\delta_{j+1} \le \delta_j/2$, the expected cost of the gathering and routing steps is at most $\sigma_{j+1}\,\mathbf{E}\,[c(T_j^*)]$. The lemma is proved. □

Putting together our bounds on the expected costs incurred in the preprocessing steps and in all of the stages of the SSBaB-Sample-Augment algorithm implies that it is a constant-factor approximation algorithm for the SSBaB problem.

Theorem 5.5. *Algorithm* SSBaB-Sample-Augment *is a 76.8-approximation algorithm for the* SSBaB *problem.*

Proof. Fix an optimal solution with cost $OPT = \sum_j C^*(j)$. By Lemmas 5.3 and 5.4, the expected cost incurred by the algorithm in stages 1 through $K-1$ is at most

$$\sum_{j=1}^{K-1} (3+\alpha)\sigma_{j+1} \cdot \mathbf{E}\,[c(T_j^*)] \;=\; (3+\alpha) \sum_{i=1}^{K} C^*(i) \cdot \left[\sum_{j=1}^{i-1} \frac{\sigma_{j+1}}{\sigma_i} + \sum_{j=i}^{K} \frac{\sigma_{j+1}}{\delta_i g_j} \right].$$

Recalling that $\sigma_{j+1}/g_j = \delta_j$ for each j and adding in the cost (13) of the preprocessing step that produces unit-weight demands for stage 1, we get that the total expected cost incurred by the SSBaB-Sample-Augment algorithm after the initial rounding of cable costs and capacities and before stage K is at most

$$(3+\alpha) \sum_{i=1}^{K} C^*(i) \cdot \left[\sum_{j=0}^{i-1} \frac{\sigma_{j+1}}{\sigma_i} + \sum_{j=i}^{K} \frac{\delta_j}{\delta_i} \right].$$

Since $\sigma_{j+1} \ge 2\sigma_j$ and $\delta_{j+1} \le \delta_j/2$ for every $j \in \{1, 2, \ldots, K-1\}$, this cost is at most $4(3+\alpha) \cdot OPT$.

In the final stage K of the algorithm, we route demands of size u_K to the sink t along shortest paths, building cables of type K to support this flow. This costs

$$\sum_{d \in D_K} \sigma_K \cdot \ell(d, t),$$

where $\ell(d,t)$ denotes the length of a shortest d-t path in G. Since every demand $d \in D_0$ with weight $w_d \leq 1$ corresponds to a demand of D_K in the final stage with probability w_d/u_K (Lemma 5.2), the expected cost of these cables of type K is

$$\sum_{d \in D_0} \frac{w_d}{u_K} \cdot \sigma_K \cdot \ell(d,t) = \delta_K \sum_{d \in D_0} w_d \ell(d,t). \tag{15}$$

Since δ_K is the smallest-possible incremental cost, the right-hand side of (15) is a lower bound on the cost of the optimal solution to $\mathcal{I}$. Thus the expected cost in the Kth stage of the SSBaB-Sample-Augment algorithm is at most OPT.

Finally, our initial rounding of the cable costs and capacities increases our approximation ratio by a factor of 4. The final approximation ratio of the SSBaB-Sample-Augment algorithm is thus $4\,[4(3+\alpha)+1]$. Using the Steiner Tree algorithm of Robins and Zelikovsky [2005], we can take $\alpha = 1.55$ to achieve an approximation ratio of 76.8. □

6. RECENT AND FUTURE WORK

We conclude by discussing recent research motivated by the present paper and some directions for future work.

6.1 Recent Work

As discussed in Section 3.2, the initial publication of our MRoB algorithm [Gupta et al. 2003] led to two subsequent papers on the problem [Becchetti et al. 2005; Fleischer et al. 2006]. The 5-approximation algorithm by Fleischer et al. [2006] is the best that is currently known for the problem.

For the SSRoB problem, Gupta et al. [2004] derandomized the algorithm given in this paper. Their approach is based on an alternative analysis of the algorithm and results in a deterministic 4.2-approximation algorithm, slightly better than the deterministic 4.55-approximation algorithm of Swamy and Kumar [2004]. Very recently, Eisenbrand et al. [2007] gave a randomized 2.92-approximation algorithm and Williamson and van Zuylen [2007] gave a deterministic 3.28-approximation algorithm for the SSRoB problem.

For buy-at-bulk problems, our SSBaB algorithm and analysis were recently refined by Grandoni and Italiano [2006] to yield a 24.92-approximation algorithm. In another direction, Charikar and Karagiozova [2005] gave the first non-trivial approximation algorithm for the generalization of the multicommodity buy-at-bulk network design problem in which the concave capacity cost function (or, equivalently, the available cable types) can vary from edge to edge. The algorithm in [Charikar and Karagiozova 2005], inspired by the Sample-Augment algorithm of this paper, randomly inflates the weight of demand pairs and then runs a greedy heuristic. Very recently, Chekuri et al. [2006] used different techniques to obtain an approximation algorithm for this problem with a polylogarithmic performance guarantee.

Two improvements of our VPND algorithm and analysis have been given recently. The first is a 4.74-approximation algorithm due to Eisenbrand and Grandoni [2005], the second a 3.55-approximation algorithm of Eisenbrand et al. [2005]. Both papers are based on variations of our algorithm and refinements of our analysis.

Most significantly, our definition of strict cost shares has been generalized and applied to give the first constant-factor approximation algorithms for several problems in *stochastic optimization*. As an example, consider the following Stochastic Steiner Tree problem. The input is a graph G with edges lengths c, a sink vertex t, a set $S = \{s_1, \ldots, s_k\}$ of sources, a distribution π over sets of sources, and an "inflation factor" $\sigma > 1$. The setup is as follows: an algorithm chooses a set F_1 of edges in the first stage; a set $\widehat{S} \subseteq S$ of sources is chosen randomly according to π; and then the algorithm chooses a set F_2 of edges so that $F_1 \cup F_2$ spans t and the sources of $\widehat{S}$. The incentive for selecting edges in the first stage, without knowledge of the realization $\widehat{S}$, is that each edge e costs c_e in the first stage but σc_e in the second stage. The goal is to design an algorithm that chooses F_1 and F_2 in a way that approximately minimizes the expectation (over π and F_2) of the total cost $c(F_1) + \sigma c(F_2)$.

Gupta et al. [2004] showed that random sampling, a Steiner Forest subroutine that admits a strengthened form of strict cost shares, and greedy augmentation can be used to obtain a 3.55-approximation algorithm for the Stochastic Steiner Tree problem. The only assumption on the distribution π in [Gupta et al. 2004] is that independent samples of π can be drawn in polynomial time. Gupta et al. [2004] also obtained similar results for stochastic versions of the Vertex Cover and Uncapacitated Facility Location problems. Earlier approximation algorithms for these problems both had weaker performance guarantees and imposed additional restrictions on the distribution π [Immorlica et al. 2004; Ravi and Sinha 2006]. Strict cost shares and generalizations have since been used to design constant-factor approximation algorithms for various other stochastic optimization problems [Gupta and Pál 2005; Gupta et al. 2005; Hayrapetyan et al. 2005].

6.2 Future Directions

We conclude the paper with several suggestions for future research.

(1) An obvious open question is to narrow the gap between the best approximation and inapproximability results for all of the problems studied in this paper. In particular, are any of these problems provably harder than the Steiner Tree problem (assuming $P \neq NP$)?

(2) A more modest goal is to understand the limitations of our analysis framework in Section 2. For example, is the guarantee in Theorem 2.10 the best possible? Is it possible to refine the definition of strict cost shares and sharpen this guarantee?

(3) Can the ideas in our MRoB and SSBaB algorithms be combined to yield an approximation algorithm for the multicommodity buy-at-bulk network design problem? While recent results of Andrews [2004] rule out constant-factor approximation algorithms under reasonable complexity assumptions, our techniques might give an $O(\log n)$-approximation algorithm for the problem that does not resort to probabilistic tree embeddings [Bartal 1998; Fakcharoenphol et al. 2004].

(4) Can the constant-factor approximation algorithm for Stochastic Steiner Tree in [Gupta et al. 2004] be extended to the stochastic version of the Steiner Forest

problem? Such an extension would follow from a strengthened version of the strict cost shares surveyed in Section 3.2.

(5) Only our SSRoB algorithm has been derandomized [Gupta et al. 2004; Williamson and van Zuylen 2007]. Can our other algorithms also be derandomized?

APPENDIX

We now describe the AKR-GW algorithm. Fix an instance $\mathcal{I} = (G, \mathcal{D})$ of Steiner Forest. For a subset $S \subseteq V$ of vertices and a demand pair (s_i, t_i), we say that S *separates* (s_i, t_i) if S contains exactly one of s_i or t_i. The set S is a *Steiner cut* of $\mathcal{I}$ if it separates some demand pair. Let $\mathcal{C}$ denote the set of Steiner cuts of $\mathcal{I}$. Finally, let $\delta(S)$ denote the set of edges with exactly one endpoint in the vertex set $S \subseteq V$. The AKR-GW algorithm iteratively constructs a feasible integral solution to the linear relaxation

$$\begin{aligned} &\min \sum_{e \in E} c_e x_e \\ &\text{subject to:} \\ (PLP) \qquad &\sum_{e \in \delta(S)} x_e \geq 1 \qquad \text{for every Steiner cut } S \in \mathcal{C} \\ &x_e \geq 0 \qquad \text{for every edge } e \in E, \end{aligned}$$

and a feasible solution to the corresponding dual linear program

$$\begin{aligned} &\max \sum_{S \in \mathcal{C}} y_S \\ &\text{subject to:} \\ (DLP) \qquad &\sum_{S \in \mathcal{C} \,:\, e \in \delta(S)} y_S \leq c_e \qquad \text{for every edge } e \in E \\ &y_S \geq 0 \qquad \text{for every Steiner cut } S \in \mathcal{C}. \end{aligned}$$

The 0-1 integer solutions to (PLP) are precisely the incidence vectors of the feasible solutions of $\mathcal{I}$. By weak linear programming duality, the objective function value of every feasible (fractional) solution to the dual program (DLP) is a lower bound on the objective function value of every feasible (fractional) solution to (PLP), and in particular on the value of a minimum-cost Steiner forest for $(G, \mathcal{D})$.

The AKR-GW algorithm maintains a set of edges, initially empty; a feasible dual solution, initially the all-zero solution; and a partition of the vertices, initially with all vertices in their own class of the partition. Edges in the current primal solution are called *tight*. We will call classes of the vertex partition *clusters*. The algorithm will maintain the invariant that clusters correspond to the connected components of the set of tight edges. A cluster is *active* if it is a Steiner cut and *inactive* otherwise.

In every iteration of the first part of the AKR-GW algorithm, the dual variables of the currently active clusters are increased by the largest common amount that does not violate any of the dual packing constraints of the form $\sum_{S \in \mathcal{C} \,:\, e \in \delta(S)} y_S \leq c_e$. After this dual increase, there is at least one edge whose packing constraint is satisfied with equality and with endpoints in different clusters, at least one of which

is active. One such edge e is then deemed *tight*, and the two clusters containing the endpoints of e are merged into a single cluster. Eventually, all clusters are inactive and this portion of the algorithm halts.

For convenience, we associate a notion of *time* with this phase of the AKR-GW algorithm. At the beginning of the algorithm the time τ is set to 0. Every time dual variables are increased, the current time increases by the same amount as the dual variables.

The final and most subtle step of the AKR-GW algorithm identifies a subset of the tight edges that is a feasible solution and also has low cost. Precisely, let F denote the set of tight edges. An edge of F is *inessential* if $F \setminus \{e\}$ is a feasible solution for $(G, \mathcal{D})$, and *essential* otherwise. The final output of the AKR-GW algorithm is the set of essential tight edges. The algorithm can clearly be implemented in polynomial time. For fast implementations, see [Cole et al. 2001; Gabow et al. 1998; Klein 1994].

Agrawal et al. [1995] and Goemans and Williamson [1995] proved the following guarantee.

THEOREM A.1 [AGRAWAL ET AL. 1995; GOEMANS AND WILLIAMSON 1995]. *For every* Steiner Forest *instance* $(G, \mathcal{D})$*, the* AKR-GW *algorithm outputs a feasible dual solution* $\{y_S\}_{S \in \mathcal{C}}$ *and a feasible Steiner forest* $F \subseteq E$ *satisfying*

$$\sum_{e \in F} c_e \leq 2 \sum_{S \in \mathcal{C}} y_S. \tag{16}$$

Since the sum on the right-hand side of (16) is a lower bound on the value of a minimum-cost Steiner forest of $(G, \mathcal{D})$, Theorem A.1 implies that the AKR-GW algorithm is a 2-approximation algorithm for the Steiner Forest problem.

ACKNOWLEDGMENTS

We thank Fabrizio Grandoni, Jochen Könemann, R. Ravi, Guido Schäfer, and the anonymous referees for helpful discussions and comments on earlier drafts of this paper.

REFERENCES

AGRAWAL, A., KLEIN, P., AND RAVI, R. 1995. When trees collide: an approximation algorithm for the generalized Steiner problem on networks. *SIAM Journal on Computing 24,* 3, 440–456. (Preliminary version in *23rd STOC*, 1991).

ANDREWS, M. 2004. Hardness of buy-at-bulk network design. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 115–124.

ANDREWS, M. AND ZHANG, L. 2002. Approximation algorithms for access network design. *Algorithmica 34,* 2, 197–215. (Preliminary version in *39th FOCS*, 1998).

ARORA, S., LUND, C., MOTWANI, R., SUDAN, M., AND SZEGEDY, M. 1998. Proof verification and the hardness of approximation problems. *Journal of the ACM 45,* 3, 501–555.

AWERBUCH, B. AND AZAR, Y. 1997. Buy-at-bulk network design. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 542–547.

AWERBUCH, B., AZAR, Y., AND BARTAL, Y. 2004. On-line generalized Steiner problem. *Theoretical Computer Science 324,* 2-3, 313–324.

BARTAL, Y. 1994. Competitive analysis of distributed on-line problems — distributed paging. Ph.D. thesis, Tel-Aviv University, Israel.

Bartal, Y. 1996. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 184–193.

Bartal, Y. 1998. On approximating arbitrary metrics by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*. 161–168.

Bartal, Y., Charikar, M., and Indyk, P. 2001. On page migration and other relaxed task systems. *Theoretical Computer Science 268,* 1, 43–66.

Bartal, Y., Fiat, A., and Rabani, Y. 1995. Competitive algorithms for distributed data management. *Journal of Computer and System Sciences 51,* 3, 341–358. (Preliminary version in *24th STOC*, 1992).

Becchetti, L., Könemann, J., Leonardi, S., and Pál, M. 2005. Sharing the cost more efficiently: Improved approximation for multicommodity rent-or-buy. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 375–384.

Bern, M. and Plassman, P. 1989. The Steiner problem with edge lengths 1 and 2. *Information Processing Letters 32,* 4, 171–176.

Charikar, M. and Karagiozova, A. 2005. On non-uniform multicommodity buy-at-bulk network design. In *Proceedings of the 37th Annual ACM Symposium on the Theory of Computing (STOC)*. 176–182.

Chekuri, C., Hajiaghayi, M. T., Kortsarz, G., and Salavatipour, M. R. 2006. Approximation algorithms for non-uniform buy-at-bulk network design problems. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 677–686.

Chekuri, C., Khanna, S., and Naor, J. 2001. A deterministic algorithm for the Cost-Distance problem. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 232–233.

Chuzhoy, J., Gupta, A., Naor, J., and Sinha, A. 2005. On the approximability of some network design problems. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 943–951.

Cole, R., Hariharan, R., Lewenstein, M., and Porat, E. 2001. A faster implementation of the Goemans-Williamson clustering algorithm. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 17–25.

Duffield, N. G., Goyal, P., Greenberg, A. G., Mishra, P. P., Ramakrishnan, K. K., and van der Merwe, J. E. 1999. A flexible model for resource management in virtual private networks. In *Proceedings of SIGCOMM*. 95–108.

Eisenbrand, F. and Grandoni, F. 2005. An improved approximation algorithm for virtual private network design. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 928–932.

Eisenbrand, F., Grandoni, F., and Rothvoss, T. 2007. A tighter analysis of random sampling for connected facility location. Submitted.

Eisenbrand, F., Grandoni, G., Oriolo, G., and Skutella, M. 2005. New approaches for virtual private network design. In *Proceedings of the 32nd Annual International Colloquium on Automata, Languages, and Programming (ICALP)*. Lecture Notes in Computer Science, vol. 3580. 1151–1162.

Fakcharoenphol, J., Rao, S., and Talwar, K. 2004. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences 69,* 3, 485–497.

Fingerhut, J. A., Suri, S., and Turner, J. S. 1997. Designing least-cost nonblocking broadband networks. *Journal of Algorithms 24,* 2, 287–309.

Fleischer, L. K., Könemann, J., Leonardi, S., and Schäfer, G. 2006. Simple cost sharing schemes for multicommodity rent-or-buy and stochastic steiner tree. In *Proceedings of the 38th Annual ACM Symposium on the Theory of Computing (STOC)*. 663–670.

Gabow, H. N., Goemans, M. X., and Williamson, D. P. 1998. An efficient approximation algorithm for the survivable network design problem. *Mathematical Programming 82,* 1-2, 13–40.

Goel, A. and Estrin, D. 2005. Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk. *Algorithmica 43,* 1-2, 5–15.

Goemans, M. X. and Williamson, D. P. 1995. A general approximation technique for constrained forest problems. *SIAM Journal on Computing 24,* 2, 296–317. (Preliminary version in *5th SODA*, 1994).

Goemans, M. X. and Williamson, D. P. 1997. The primal-dual method for approximation algorithms and its application to network design problems. In *Approximation Algorithms for NP-hard Problems*, D. S. Hochbaum, Ed. PWS Publishing.

Grandoni, F. and Italiano, G. F. 2006. Improved approximation for single-sink buy-at-bulk. In *17th International Symposium on Algorithms and Computation (ISAAC)*. 111–120.

Guha, S., Meyerson, A., and Munagala, K. 2000. Hierarchical placement and network design problems. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 603–612.

Guha, S., Meyerson, A., and Munagala, K. 2001. 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)*. 383–388.

Gupta, A., Hajiaghayi, M. T., and Räcke, H. 2006. Oblivious network design. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 970–979.

Gupta, A., Kumar, A., Kleinberg, J., Rastogi, R., and Yener, B. 2001. Provisioning a Virtual Private Network: A network design problem for multicommodity flow. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*. 389–398.

Gupta, A., Kumar, A., Pál, M., and Roughgarden, T. 2003. Approximation via cost-sharing: A simple approximation algorithm for the multicommodity rent-or-buy problem. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 606–615.

Gupta, A., Kumar, A., and Roughgarden, T. 2003. Simpler and better approximation algorithms for network design. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*. 365–372.

Gupta, A. and Pál, M. 2005. Stochastic Steiner trees without a root. In *Proceedings of the 32nd Annual International Colloquium on Automata, Languages, and Programming (ICALP)*. Lecture Notes in Computer Science, vol. 3580. 1051–1063.

Gupta, A., Pál, M., Ravi, R., and Sinha, A. 2004. Boosted sampling: Approximation algorithms for stochastic optimization. In *Proceedings of the 36th Annual ACM Symposium on the Theory of Computing (STOC)*. 417–426.

Gupta, A., Pál, M., Ravi, R., and Sinha, A. 2005. What about Wednesday? Approximation algorithms for multistage stochastic optimization. In *Proceedings of the 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*. Lecture Notes in Computer Science, vol. 3624. 86–98.

Gupta, A., Srinivasan, A., and Tardos, É. 2004. Cost-sharing mechanisms for network design. In *Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*. Lecture Notes in Computer Science, vol. 3122. 139–150.

Hassin, R., Ravi, R., and Salman, F. S. 2004. Approximation algorithms for a capacitated network design problem. *Algorithmica 38,* 3, 417–431.

Hayrapetyan, A., Swamy, C., and Tardos, É. 2005. Network design for information networks. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 933–942.

Immorlica, N., Karger, D. R., Minkoff, M., and Mirrokni, V. S. 2004. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 691–700.

Jain, K. and Vazirani, V. 2001. Applications of approximation algorithms to cooperative games. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing (STOC)*. 364–372.

Jain, K. and Vazirani, V. 2002. Equitable cost allocations via primal-dual-type algorithms. In *Proceedings of the 34th Annual ACM Symposium on the Theory of Computing (STOC)*. 313–321.

Karger, D. R. and Minkoff, M. 2000. Building Steiner trees with incomplete global knowledge. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 613–623.

Khuller, S. and Zhu, A. 2002. The general Steiner tree-star problem. *Information Processing Letters 84,* 4, 215–220.

Kim, T. U., Lowe, T. J., Tamir, A., and Ward, J. E. 1996. On the location of a tree-shaped facility. *Networks 28,* 3, 167–175.

Klein, P. N. 1994. A data structure for bicategories, with application to speeding up an approximation algorithm. *Information Processing Letters 52,* 6, 303–307.

Könemann, J., Leonardi, S., and Schäfer, G. 2005. A group-strategyproof mechanism for Steiner forests. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 612–619.

Kumar, A., Gupta, A., and Roughgarden, T. 2002. 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 (FOCS)*. 333–342.

Labbé, M., Laporte, G., Martín, I. M., and Salazar González, J. J. 2001. The median cycle problem. Tech. Rep. 2001/12, Department of Operations Research and Multicriteria Decision Aid at Université Libre de Bruxelles.

Lee, Y., Chiu, Y., and Ryan, J. 1996. A branch and cut algorithm for a Steiner tree-star problem. *INFORMS Journal on Computing 8,* 3, 194–201.

Meyerson, A., Munagala, K., and Plotkin, S. 2000. Cost-Distance: Two metric network design. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 624–630.

Meyerson, A., Munagala, K., and Plotkin, S. 2001. Designing networks incrementally. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 406–415.

Pál, M. 2004. Cost sharing and approximation. Ph.D. thesis, Cornell University.

Pál, M. and Tardos, É. 2003. Group strategyproof mechanisms via primal-dual algorithms. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 584–593.

Ravi, R. and Salman, F. S. 1999. Approximation algorithms for the traveling purchaser problem and its variants in network design. In *Proceedings of the 7th Annual European Symposium on Algorithms (ESA)*. Lecture Notes in Computer Science, vol. 1643. 29–40.

Ravi, R. and Sinha, A. 2006. Hedging uncertainty: Approximation algorithms for stochastic optimiza tion problems. *Mathematical Programming 108,* 1, 97–114.

Robins, G. and Zelikovsky, A. 2005. Tighter bounds for graph Steiner tree approximation. *SIAM Journal on Discrete Mathematics 19,* 1, 122–134.

Salman, F. S., Cheriyan, J., Ravi, R., and Subramanian, S. 2000. Approximating the single-sink link-installation problem in network design. *SIAM Journal on Optimization 11,* 3, 595–610. (Preliminary version in *8th SODA*, 1997).

Swamy, C. and Kumar, A. 2004. Primal-dual algorithms for connected facility location problems. *Algorithmica 40,* 4, 245–269.

Talwar, K. 2002. Single-sink buy-at-bulk LP has constant integrality gap. In *Proceedings of the 9th Integer Programming and Combinatorial Optimization Conference (IPCO)*. Lecture Notes in Computer Science, vol. 2337. 475–486.

Vazirani, V. V. 2001. *Approximation Algorithms*. Springer-Verlag, Berlin.

Williamson, D. P. and van Zuylen, A. 2007. A simpler and better derandomization of an approximation algorithm for single-source rent-or-buy. *Operations Research Letters*. To appear.

Young, H. P. 1994. Cost allocation. In *Handbook of Game Theory*, R. J. Aumann and S. Hart, Eds. Vol. 2. North-Holland, Chapter 34, 1193–1235.

Received Month Year; revised Month Year; accepted Month Year