THEORY OF COMPUTING, Volume 8 (2012), pp. 351–368
www.theoryofcomputing.org

SPECIAL ISSUE IN HONOR OF RAJEEV MOTWANI

One Tree Suffices: A Simultaneous $O(1)$-Approximation for Single-Sink Buy-at-Bulk

Ashish Goel* Ian Post†

Received: September 14, 2010; published: July 20, 2012.

Abstract: We study the single-sink buy-at-bulk problem with an unknown cost function. We wish to route flow from a set of demand nodes to a root node, where the cost of routing x total flow along an edge is proportional to $f(x)$ for some concave, non-decreasing function f satisfying $f(0) = 0$. We present a simple, fast, combinatorial algorithm that takes a set of demands and constructs a single tree T such that for all f the cost $f(T)$ is a 47.45-approximation of the optimal cost for that particular f. This is within a factor of 2.33 of the best approximation ratio currently achievable when the tree can be optimized for a specific function. Trees achieving simultaneous $O(1)$-approximations for all concave functions were previously not known to exist regardless of computation time.

ACM Classification: F.2.2, G.2.2

AMS Classification: 68W25, 68R10

Key words and phrases: network design, buy-at-bulk, simultaneous optimization

1 Introduction

Many natural network design settings exhibit some form of economies of scale that reduce the costs when many flows are aggregated together. We may benefit from cheaper bandwidth when laying high

*Research funded by an NSF IIS grant, by funds from Google, Microsoft, and Cisco, a 3-COM faculty fellowship, and by the ARL Network Science CTA.

†Research funded by an NSF IIS grant.

 DOI: 10.4086/toc.2012.v008a015

capacity network links [2], reduced infrastructure costs or bulk discounts for shipping large amounts of goods together [29, 30], or summarization and compression of correlated information flows [26]. These scenarios are known in the literature as buy-at-bulk problems. In a general buy-at-bulk problem we are given a graph and a set of demands for flow between nodes. The cost of routing a total of x_e flow along an edge e with length ℓ_e is $\ell_e f(x_e)$ for some function f, and the goal is to find a collection of routes satisfying the demands that minimizes the total cost $\sum_e \ell_e f(x_e)$. To model the economies of scale, we assume f is concave and monotone non-decreasing.

We will focus on single-sink (or single-source) case, where all demands must be routed to a given root. When f is known, the problem becomes the well-studied single-sink buy-at-bulk (SSBaB) problem. SSBaB is NP-hard—it generalizes the Steiner tree problem—but constant-factor approximations are known for any given f (e. g., [17, 18, 16]). The special case where f has the form $f(x) = \min\{x, M\}$ for some M (edges can be "rented" for linear cost or "bought" for a fixed cost) is known as the single-sink rent-or-buy (SSRoB) problem and has also received significant attention (e. g., [24, 7]).

Buy-at-bulk algorithms produce trees that are heavily tailored to the particular function at hand, but in some scenarios f may be unknown or known to change over time. One setting where this arises is in aggregation of data in sensor networks. The degree of redundancy among different sensor measurements may be unknown, or the same network may be used for aggregating different types of information with different amounts of redundancy. In other situations rapid technological advancement may cause bandwidth costs to change drastically over time. Further, in the interest of simplifying the design process and building robust networks, it may be useful to decouple the problem of designing the network topology from that of determining the exact characteristics of the information or goods flowing through that network. In these settings it is desirable to find a single tree that is simultaneously good for all cost-functions, and from a theoretical perspective, the existence of such trees would reveal surprising structure in the problem.

There are two natural objective functions which capture the idea of simultaneous approximation for multiple cost functions. Let $\mathcal{F}$ be the set of all concave, monotone non-decreasing cost functions f satisfying $f(0) = 0$ and T^*_f be the optimal routing graph for f. Note that due to the concavity of f, we may assume that T^*_f is a tree (Lemma 1 in [2]). For a routing tree T, we use the shorthand $f(T)$ to denote the cost of T under function f. Formally, $f(T) = \sum_e \ell_e f(x_e)$, where x_e is the amount of flow T sends on edge e. Let $\mathcal{R}$ be a randomized algorithm that returns a feasible routing tree T. First, we could try to minimize the quantity

$$\sup_{f \in \mathcal{F}} \frac{\mathbf{E}_{\mathcal{R}}[f(T)]}{f(T^*_f)} \tag{1.1}$$

which we call the *oblivious* approximation ratio. If the oblivious ratio is small, then $\mathcal{R}$ returns a distribution that works well in expectation for any f. However, there may be no sample from this distribution that works for everything: for any tree T there may be functions for which $f(T)$ is expensive.

To circumvent this problem we can work with the much stronger *simultaneous* approximation ratio. For a deterministic or randomized algorithm $\mathcal{A}$ that returns a tree $T_{\mathcal{A}}$ the simultaneous ratio of $\mathcal{A}$ is defined as:

$$\mathbf{E}_{\mathcal{A}}\left[\sup_{f \in \mathcal{F}} \frac{f(T_{\mathcal{A}})}{f(T^*_f)}\right] \tag{1.2}$$

A bound on the simultaneous ratio subsumes one on the oblivious ratio and proves there exists a single tree that is simultaneously good for all f.

We emphasize that the distinction between the simultaneous and oblivious objectives is not a technicality in the objective but rather a fundamental difference and that the gap between these ratios can be large. Consider the problem of embedding arbitrary metrics into tree metrics, another problem that requires bounding the cost under many different functions (i. e., distortion of each edge). It is well-known that distributions over trees can achieve $O(\log n)$ expected distortion for all edges [10] but that even for simple graphs like the n-cycle no single tree can do better than $\Omega(n)$ distortion [27]. Therefore, the ratio between maximum expected distortion and the expected maximum distortion is $\Omega(n/\log n)$ in this case.

Goel and Estrin [12] introduced the problem of simultaneous SSBaB and gave an algorithm with an $O(\log D)$ bound on the simultaneous ratio (1.2), where D is the total amount of demand. Prior to our work,[1] it was not known whether a simultaneous ratio (or even the weaker oblivious ratio) of $O(1)$ could be achieved regardless of computation time. In this paper we give the first constant guarantee on the simultaneous ratio, resolving the major open question of Goel and Estrin and Goel and Post [12, 13].

Several aspects of our algorithm and analysis bear mentioning. First, given a deterministic λ-approximation for SSRoB, we achieve a simultaneous ratio of $(1+\varepsilon)(8+4\sqrt{5})\lambda$, and we achieve this ratio in expectation if the approximation is randomized. This results in a deterministic algorithm with a simultaneous ratio of 55.58, and using a randomized SSRoB approximation, our simultaneous ratio improves to 47.45 in expectation, which is within a factor of 2.33 of the current best approximation for normal SSBaB of 20.42 [16]. Second, the algorithm is entirely combinatorial, and our analysis is short and simple, no more complex than the analysis of a normal SSBaB algorithm. Third, using a deterministic SSRoB approximation with a runtime of $t(n,m)$ for a graph with n nodes and m edges, our runtime is only $O((t(n,m)+m+n\log n)\log D)$ where D is the total demand. If the SSRoB algorithm is randomized, the runtime increases to $O((t(n,m)\log\log D+m+n\log n)\log D)$.

The algorithm is no more complex than one for SSBaB, and at a high level, if not in the details, it follows the template of many SSBaB algorithms. We first find approximate trees for a set of rent-or-buy basis functions and divide the edges of each tree into rent and buy costs based on whether the rent-or-buy function has leveled off on an edge. We prune this set to obtain a subset L of trees and corresponding cost functions whose total rent costs are increasing geometrically while total buy costs are dropping geometrically, and then prove it suffices to simultaneously approximate only the functions in L. Each tree in L has a set of core nodes spanned by edges that are bought, and these cores define a series of tree layers, which we stitch together using light approximate shortest-path trees (LASTs) [25] to approximate both the minimum spanning tree (MST) and shortest-path tree. Finally, we consider any layer in the tree. Using the geometrically changing costs and the properties of the LAST construction, we conclude that everything within the layer is an approximate MST, and everything outside approximates the shortest-path tree cost.

[1] This submission combines two conference publications, in which we reduced first the oblivious ratio [13] and subsequently the simultaneous ratio [14] to $O(1)$; our exposition follows the latter of these two publications.

1.1 Related work

The SSBaB problem was first posed by Salman et al. [29, 30], and the first general approximation algorithm was given by Awerbuch and Azar [3], who used metric tree embeddings [4] to achieve an $O(\log^2 n)$ ratio, later improved to $O(\log n)$ using better embeddings by Bartal [5] and Fakcharoenphol et al. [10]. Guha et al. [17, 18] gave the first constant approximation, and follow-up work by Talwar [32], Gupta et al. [21], Jothi and Raghavachari [23], Grandoni and Italiano [15], and Grandoni and Rothvoß [16] has since reduced the constant to 20.42. Most recent algorithms for SSBaB (and several related problems) are based on the sample and augment framework of Gupta et al. [21]. Many algorithms using this framework have been derandomized by van Zuylen [33].

The special case of SSRoB has also been extensively studied, often as a special case of the connected facility location problem. The first constant factor approximation was given by Ravi and Salman [28] as a special case of the traveling purchaser problem. Karger and Minkoff [24] gave an alternate algorithm and introduced connected facility location. Gupta et al. improved the approximation to 9.01 [20], Swamy and Kumar to 4.55 [31], and Gupta et al. to 3.55 [21]. Gupta et al. [22] derandomized the 3.55-approximation to achieve a 4.2-approximation. Eisenbrand et al. [7] developed a randomized 2.92-approximation, which recently improved to 2.8 using the 1.39-approximation for Steiner tree of Byrka et al. [6]. Since we employ the 2.8-approximation, and the claimed ratio does not currently appear elsewhere in the literature, we present the brief calculation deriving this value in Section 5. Both Williamson and van Zuylen [34] and Eisenbrand et al. [7] independently derandomized the 2.92-approximation to achieve a deterministic 3.28-approximation.

The problem of simultaneous approximation for multiple cost functions has been studied using both the oblivious and simultaneous objectives. Goel and Estrin [12] were the first to explicitly pose the question of simultaneous approximations and gave an algorithm with an $O(\log D)$ simultaneous guarantee. Prior to that Khuller et al. [25] gave an algorithm to simultaneously approximate the two extreme cost functions $f(x) = 1$ and $f(x) = x$—a result which plays an important role in this paper—and metric tree embeddings had been applied to SSBaB [3, 10] to achieve an $O(\log n)$ bound for the oblivious objective. Enachescu et al. [8] gave an $O(1)$ simultaneous guarantee for the special case of grid graphs with some spatial correlation. Gupta et al. [19] and Englert and Räcke [9] have studied several generalizations of the problem where both the demands and function are unknown, and multiple sinks are allowed. In these settings the guarantee is generally $O(\log n)$ or $O(\text{polylog } n)$.

2 Notation and preliminaries

Formally, we are given a graph $G = (V,E)$ with edge lengths ℓ_e for $e \in E$, a root node r, and a set of demand nodes $\mathcal{D} \subseteq V$ with integer demands d_v. The total demand is $D = \sum_v d_v$. We want to route d_v flow from each v to r as cheaply as possible, where the cost of routing x_e flow along edge e is $\ell_e f(x_e)$ for some unknown, concave, monotone increasing function f satisfying $f(0) = 0$. Not knowing f, our objective is to find a feasible tree T minimizing $\sup_f f(T)/f(T_f^*)$, where T_f^* is the optimal graph for f.

We first show that we can restrict our analysis to a smaller class of basis functions, a technique originally introduced by Goel and Estrin [12]. Let $\varepsilon > 0$ be a small constant that will trade off the runtime and the approximation ratio, and $K = \lceil \log_{1+\varepsilon} D \rceil$. For $0 \le i \le K$, define $M_i = (1+\varepsilon)^i$, $A_i(x) = \min\{x, M_i\}$, and T_i^* as the optimal tree for A_i. By the monotonicity and concavity of f, whenever $M_i \le x \le M_{i+1}$ we have $f(M_i) \le f(x) \le f(M_{i+1}) \le (1+\varepsilon)f(M_i)$, so with a loss of only a factor of $1+\varepsilon$ we can interpolate between $f(M_i)$ and $f(M_{i+1})$ and assume f is piecewise linear with breakpoints only at powers of $1+\varepsilon$. Such a piecewise linear function can be written as a nonnegative linear combination $f(x) = \sum_i a_i A_i(x)$ of $\{A_i\}_{0\le i\le K}$ by setting coefficients a_i equal to the changes in slope: if the slope drops from δ_i to δ_{i+1} at $(1+\varepsilon)^i$ it induces the term $(\delta_i - \delta_{i+1})A_i(x)$. Now for a linear combination $\sum_i a_i A_i(x)$ and a tree T

$$\frac{\sum_i a_i A_i(T)}{\sum_i a_i A_i(T_f^*)} \le \frac{\sum_i a_i A_i(T)}{\sum_i a_i A_i(T_i^*)} = \frac{\sum_i a_i A_i(T_i^*)\frac{A_i(T)}{A_i(T_i^*)}}{\sum_i a_i A_i(T_i^*)} \le \max_i \frac{A_i(T)}{A_i(T_i^*)}$$

so it suffices to upper bound $\max_i A_i(T)/A_i(T_i^*)$.

We now define some notation and subroutines that will be important for our algorithm. The problem of finding a good aggregation tree for the function $A_i(x) = \min\{x, M_i\}$ is an instance of the SSRoB problem, and we can find a λ-approximate tree T_i, where λ is the best approximation ratio known, currently equal to 2.8 using the algorithm of Eisenbrand et al. [7] and Byrka et al. [6]. In the case of randomized algorithms, such as the current 2.8-approximation, we run the approximation $O(\log\log D)$ times for each value of i and pick the best output, which ensures that the probability of any of the $O(\log D)$ trees T_i being worse than a $(1+\varepsilon)\lambda$-approximation is at most a constant. The remainder of our algorithm is entirely deterministic, so this is the only possibility of failure. To ensure a totally deterministic algorithm we can alternately use the 3.28-approximation algorithm of Williams and van Zuylen [34] and Eisenbrand et al. [7].

The cost $A_i(T_i)$ can be broken into two pieces, the rent cost and the buy cost, based on whether A_i is maxed out at M_i:

Definition 2.1. For an aggregation tree T_i for cost function A_i with x_e flow on edge e, the *rent cost* R_i and *normalized buy cost* B_i are defined as

$$R_i = \sum_{e\in T_i, x_e < M_i} \ell_e A_i(x_e)\,,$$

$$B_i = \sum_{e\in T_i, x_e \ge M_i} \ell_e \frac{A_i(x_e)}{M_i} = \sum_{e\in T_i, x_e\ge M_i} \ell_e\,.$$

Note that edges composing R_i are scaled by the costs incurred, but edges in B_i are not; they use unscaled edge lengths. The total cost of T_i is given by $A_i(T_i) = R_i + M_i B_i$. The rent and buy costs also partition the nodes of T_i into two sets:

Definition 2.2. The *core* C_i of tree T_i consists of r and all nodes spanned by bought edges—edges carrying at least M_i flow—and the *periphery* contains all vertices outside C_i.

The core C_i forms a connected component of T_i, and if we condition on the nodes in C_i then the rent-or-buy problem becomes easy: demands outside the core pay linear cost until they reach C_i, so they should take the shortest path, whereas within C_i we pay a fixed cost per edge length, so the best strategy is to follow the min spanning tree. The cost R_i is therefore at least the sum of shortest path distances to C_i, while B_i is at least the weight of the MST of C_i.

In addition to the SSRoB approximation, we will also employ the light, approximate shortest-path tree algorithm of Khuller et al. [25]:

Definition 2.3 ([25])**.** For $\alpha \geq 1$ and $\beta \geq 1$, an (α, β)-*light, approximate shortest-path* or (α, β)-*LAST* is a spanning tree T of G with root r such that

- for each vertex v, the distance from v to r in T is at most α times the shortest path distance from v to r in G, and
- the edge weight of T is at most β times the weight of an MST of G.

Khuller et al. show how to construct an (α, β)-LAST for any $\alpha > 1$ and $\beta \geq (\alpha+1)/(\alpha-1)$. Roughly, the algorithm performs a depth-first traversal of the MST of G starting from r, checking the stretch of the shortest path to each node. If the path to some v has blown up by at least an α factor, then it updates the tree to take the shortest path from v to r, adjusting other tree edges and distances accordingly. See the paper [25] for a full description and analysis.

Finally, we define four parameters α, β, γ, and δ used by our algorithm whose values we will optimize at the end:

- $\alpha > 1$ is the approximation ratio for shortest paths used in our LAST,
- $\beta \geq (\alpha+1)/(\alpha-1)$ is the corresponding approximation to the MST in the LAST,
- $\gamma > 1$ is the factor by which we require normalized buy costs B_i to drop as i increases, moving inward layer by layer in our tree, and
- $\delta > 1$ is the factor by which rent costs R_i grow as i increases from layer to layer in our tree.

We now turn to a more thorough explanation of tree layers.

3 Tree layers

In the normal SSBaB problem when $f(x)$ is given, the cost function is defined as $f(x) = \min_j\{\sigma_j + \delta_j x\}$, the cheapest of a collection of different "pipes" or "cables" given to the algorithm, each with an affine cost function $\sigma_j + \delta_j x$. It is common (e. g., [17, 18, 21]) to first prune these pipes to a smaller set with geometrically decreasing δ_j's and geometrically increasing σ_j's and then build a solution in layers where each layer routes with a different pipe.

We perform an analogous procedure. We would like to build our simultaneous tree T in a series of nested layers defined by the cores C_i of each tree T_i, so that the core of T under A_i is similar to C_i, but we have no guarantees on the relationships between different cores: C_i and C_k may be entirely disjoint except for r. However, we will show that as long as normalized buy costs B_i and B_k are within a constant factor of each other, the same core can be used for both trees. Consequently, we are able to define nested layers by choosing one C_i for each order of magnitude of B_i.

After finding λ-approximate trees T_i for each A_i, we loop through the costs B_i and R_i, discarding i whenever B_i does not drop by γ or R_i does not grow by δ. We are left with a subset L of the T_i where the B_i's are dropping by a factor of γ and the R_i's are growing by a factor of δ. The cores C_i for each $i \in L$ will define the layers of our tree. Algorithm 1 describes the procedure in more detail.

Algorithm 1: Finding tree layers

Input: Graph G and demands $\mathcal{D}$
Output: Set L and cores C_i for each $i \in L$

1 **for** $i \leftarrow 0$ **to** K **do**
2     $T_i \leftarrow \lambda$-approximate tree for $A_i(x)$
3 **for** $i \leftarrow 1$ **to** K **do**
4   **if** $A_i(T_{i-1}) < A_i(T_i)$ **then** $T_i \leftarrow T_{i-1}$
5 **for** $i \leftarrow K-1$ **down to** 0 **do**
6   **if** $A_i(T_{i+1}) < A_i(T_i)$ **then** $T_i \leftarrow T_{i+1}$
7 **for** $i \leftarrow 0$ **to** K **do**
8   calculate C_i, B_i, R_i
9 $L_B \leftarrow \emptyset$
10 $B \leftarrow \infty$
11 **for** $i \leftarrow 0$ **to** K **do**
12   **if** $B_i < \frac{1}{\gamma}B$ **then**
13         $L_B \leftarrow L_B \cup \{i\}$
14         $B \leftarrow B_i$
15 $L \leftarrow \emptyset$
16 $R \leftarrow \infty$
17 **foreach** $i \in L_B$ *in decreasing order* **do**
18   **if** $R_i < \frac{1}{\delta}R$ **then**
19         $L \leftarrow L \cup \{i\}$
20         $R \leftarrow R_i$
21 **return** *L and C_i for each $i \in L$*

Intuitively, as it becomes more expensive to buy edges the optimum will buy fewer edges and rent more. In the case of approximations, the progression becomes muddled because for some i the approximation guarantee may be tight, while for $i+1$ we may get lucky and find the optimum, resulting in both rent and normalized buy costs dropping. We first show that the monotonicity in buy and rent costs still holds as long as each T_i is better for A_i than both T_{i-1} and T_{i+1}.

Lemma 3.1. *After line 8 of Algorithm 1, for every i we have $B_i \geq B_{i+1}$ and $R_i \leq R_{i+1}$.*

Proof. First we show that for each i, $A_i(T_i) \leq \min\{A_i(T_{i+1}), A_i(T_{i-1})\}$. After the loop on lines 3–4 we have $A_i(T_i) \leq A_i(T_{i-1})$, and after the second loop on lines 5–6 we have $A_i(T_i) \leq A_i(T_{i+1})$, so we only need to show that the second loop does not break the first condition. If the second loop updates T_i, then $A_i(T_i)$ will only shrink, and if it changes T_{i-1} it does this by setting $T_{i-1} \leftarrow T_i$, which preserves $A_i(T_i) \leq A_i(T_{i-1})$.

Now consider $A_i(T_k)$ for any k. By definition $A_i(x) \leq x$ and $A_i(x) \leq M_i$, so to upper bound $A_i(T_k)$ we may assume edges within C_k pay M_i per unit length, which sums to M_iB_k, and edges outside C_k pay linear cost, or R_k total, implying $A_i(T_k) \leq R_k + M_iB_k$. Therefore

$$R_i + M_iB_i = A_i(T_i) \leq A_i(T_{i+1}) \leq R_{i+1} + M_iB_{i+1} \Longrightarrow M_i(B_i - B_{i+1}) \leq R_{i+1} - R_i\,.$$

Similarly,

$$R_{i+1}+M_{i+1}B_{i+1}=A_{i+1}(T_{i+1})\leq A_{i+1}(T_i)\leq R_i+M_{i+1}B_i \Longrightarrow R_{i+1}-R_i\leq M_{i+1}(B_i-B_{i+1}).$$

Combining the inequalities,

$$M_i(B_i-B_{i+1})\leq R_{i+1}-R_i\leq M_{i+1}(B_i-B_{i+1}).$$

If $B_i-B_{i+1}<0$ the inequality is false because $M_{i+1}>M_i$, so we conclude $B_i\geq B_{i+1}$. And using the first inequality, $0\leq M_i(B_i-B_{i+1})\leq R_{i+1}-R_i$, so $R_i\leq R_{i+1}$. □

We need to show that we can restrict our attention to T_i for $i\in L$. Suppose $i<k$ but $B_i\leq\gamma B_k$. Using Lemma 3.1, observe that

$$A_k(T_i)\leq R_i+M_kB_i\leq R_k+\gamma M_kB_k\leq\gamma A_k(T_k).$$

Note this is independent of the size of the intersection of the cores C_i and C_k and any differences in routing. The following lemma generalizes this simple but key observation and proves that approximating each $i\in L$ is sufficient.

Lemma 3.2. *Suppose there exists a tree T and constants c_B and c_R such that for all $i\in L$ there exists a partition of the edges of T into two sets T_{B_i} and T_{R_i} satisfying*

- $A_0(T_{B_i})\leq c_BB_i$ *and*
- $A_K(T_{R_i})\leq c_RR_i$,

then for all $k\in\{0,\dots,K\}$, $A_k(T)\leq\max\{c_B\gamma,c_R\delta\}\lambda A_k(T_k^*)$.

Proof. Let $k\in\{0,\dots,K\}$. Let $j=\max\{j\in L_B\mid j\leq k\}$. Either $j=k$, or k was discarded due to j on lines 11–14 because $B_k\geq(1/\gamma)B_j$, and either way $B_j\leq\gamma B_k$. Now let $i=\min\{i\in L\mid i\geq j\}$. Again, either $i=j$, or j was pruned due to i on lines 17–20, implying $R_i\leq\delta R_j$ (as well as $i>k$). Applying Lemma 3.1 with $i\geq j$ and $j\leq k$, we have $B_i\leq B_j\leq\gamma B_k$ and $R_i\leq\delta R_j\leq\delta R_k$.

This is sufficient to bound the cost of $A_k(T)$:

$$\begin{aligned}A_k(T)=A_k(T_{R_i})+A_k(T_{B_i})&\leq A_K(T_{R_i})+M_kA_0(T_{B_i})\\&\leq c_RR_i+c_BM_kB_i\\&\leq c_R\delta R_k+c_B\gamma M_kB_k\\&\leq\max\{c_R\delta,c_B\gamma\}A_k(T_k)\leq\max\{c_R\delta,c_B\gamma\}\lambda A_k(T_k^*).\end{aligned}$$

The equality follows because T_{B_i} and T_{R_i} partition the edges of T. The first inequality holds because $A_K(x)$ and $M_kA_0(x)$ both upper bound $A_k(x)$, the second is by assumption, the third is from the derivation above, the fourth applies $A_k(T_k)=R_k+M_kB_k$, and the last uses that T_k is a λ-approximation. □

We will primarily assume that our SSRoB algorithm is a generic approximation, but Lemma 3.2 can easily be improved to take advantage of an SSRoB algorithm with a stronger guarantee that separately bounds R_i and M_iB_i in terms of the optimal costs R_i^* and $M_iB_i^*$ as in [7].

Corollary 3.3. *Let T, c_B, and c_R be as in Lemma 3.2, and suppose*

- $R_i \leq \mu_R R_i^* + \mu_B M_i B_i^*$ *and*
- $M_i B_i \leq \nu_R R_i^* + \nu_B M_i B_i^*$,

then for all k, $A_k(T) \leq \max\{c_R\delta\mu_R + c_B\gamma\nu_R, c_R\delta\mu_B + c_B\gamma\nu_B\} A_k(T_k^*)$.

Proof. We change the inequalities in the proof above as follows:

$$\begin{aligned} A_k(T) \leq c_R\delta R_k + c_B\gamma M_k B_k &\leq c_R\delta(\mu_R R_k^* + \mu_B M_k B_k^*) + c_B\gamma(\nu_R R_k^* + \nu_B M_k B_k^*) \\ &\leq \max\{c_R\delta\mu_R + c_B\gamma\nu_R, c_R\delta\mu_B + c_B\gamma\nu_B\} A_k(T_k^*) \,. \end{aligned}$$ □

4 Constructing the tree

The construction of the tree itself is quite simple. We have a set of indices L and core sets C_i for $i \in L$. Starting with the largest $i \in L$, i. e., smallest B_i, and working downward, we connect each C_i to T—the tree so far—with a LAST. Algorithm 2 describes the procedure more formally. The notation G/T represents contracting T to a single node in G, and $G[C_i]$ is the induced subgraph on C_i, so $(G/T)[C_i]$ denotes first contracting T and then restricting to nodes in C_i.

Algorithm 2: Constructing the tree

Input: Graph G, set L, and accompanying C_i for each $i \in L$
Output: Aggregation tree T

$T \leftarrow \{r\}$
foreach $i \in L$ *in decreasing order* **do**
  $T' \leftarrow (\alpha, \beta)$-LAST of $(G/T)[C_i]$ with root T
  $T \leftarrow T \cup T'$
return T

Lemma 4.1. *The graph T constructed by Algorithm 2 is a tree and spans all demand nodes.*

Proof. Observe that $0 \in L$, and $R_0 = 0$, so C_0 covers all demands. Therefore after the last iteration T spans $\mathcal{D}$. Each iteration only adds edges spanning new vertices, so no cycles are created. □

The tree may contain paths connecting Steiner nodes that carry no flow. Such edges can be safely pruned or just ignored because they contribute nothing to the cost.

All that remains is to define the partitions T_{B_i} and T_{R_i} and prove the bounds needed in Lemma 3.2. The set T_{B_i} contains all edges present in T after connecting C_i, and T_{R_i} contains the rest. Both cost bounds will follow easily from the geometrically changing costs: the cost $A_0(T_{B_i})$ is dominated by the cost of the C_i layer, an approximate MST, and $A_K(T_{R_i})$ is dominated by the rent costs of the next layer, an approximate shortest-path tree. First, we bound the normalized buy cost of T_{B_i}.

Lemma 4.2. *Let $i \in L$, and T_{B_i} be the tree constructed by Algorithm 2 after the round when C_i is added. Then the edge cost $A_0(T_{B_i})$ is at most*

$$\frac{\beta\gamma}{\gamma-1} B_i \,.$$

Proof. We will prove by decreasing induction on i, i. e., the order in which the layers are built, that $A_0(T_{B_i}) \le cB_i$ for some c to be determined below. The base case corresponds to the first tree layer, which is indexed by the largest i in L, or, equivalently, the smallest i such that $B_i = 0$. In this case, $C_i = \{r\}$, $T_{B_i} = \{r\}$, and the edge cost is 0.

Now let $i \in L$, $k = \min\{k \in L \mid k > i\}$ be the previous (inner) layer, and suppose the edge cost of T_{B_k} is at most cB_k. By the construction of L, we know $B_k < (1/\gamma)B_i$, implying T_{B_k} costs at most $(c/\gamma)B_i$. The cost of an MST of C_i in G is at most B_i, and T_{B_k} may already span part of C_i, so connecting the rest with an MST[2] of $(G/T_{B_k})[C_i]$ costs at most B_i. Using an (α, β)-LAST scales the cost by at most β.

The total cost of edges laid so far is at most $(c/\gamma)B_i + \beta B_i$, so the proof is complete as long as $cB_i \ge (c/\gamma)B_i + \beta B_i$. Set $c = \beta\gamma/(\gamma-1)$; then we conclude that

$$c \ge \beta + \frac{c}{\gamma} \iff c\left(1 - \frac{1}{\gamma}\right) \ge \beta \iff c \ge \frac{\beta\gamma}{\gamma-1} \,.$$

□

Now we bound the rent costs of T_{R_i} by a similar proof.

Lemma 4.3. *Let $i \in L$, T be the final tree output by Algorithm 2, and $T_{R_i} = T/T_{B_i}$, i. e., all edges outside of T_{B_i}. Then the rent cost $A_K(T_{R_i})$ is at most*

$$\frac{\alpha\delta}{\delta-\alpha-1} R_i \,.$$

Proof. We prove by increasing induction on $i \in L$ (the reverse of Lemma 4.2) that $A_K(T_{R_i}) \le cR_i$ for some c to be determined. Since $0 \in L$, and T_{B_0} covers everything, the base case T_{R_0} costs 0 too.

For the inductive case, let $i \in L$, $k = \max\{k \in L \mid k < i\}$ be the next (outer) layer, and T_{R_k} have rent cost at most cR_k under A_K. As before, note $R_k < (1/\delta)R_i$, so $A_K(T_{R_k}) \le (c/\delta)R_i$. Tree T_{B_i} spans C_i and possibly more, so if all demands outside T_{B_i} take the shortest path from their sources to T_{B_i}, then the shortest-path cost is at most R_i. However, consider the path taken through T from a source node s to T_{B_i}. The node where the path transitions from edges of T_{R_k} to T_{B_k} may actually be farther away from T_{B_i} than s is. But by the triangle inequality, the cost of sending all demands from the point they leave T_{R_k} to T_{B_i} via shortest paths is at most $(c/\delta)R_i + R_i$, the cost of sending all flow in T_{R_k} back to its source and from there to T_{B_i} using shortest paths. The LAST algorithm guarantees α-approximate shortest paths, multiplying the cost by α.

Consequently, the total rent cost for T_{R_i} is bounded by $(c/\delta)R_i + \alpha((c/\delta)R_i + R_i)$, which needs to be at most cR_i. Set $c = \alpha\delta/(\delta-\alpha-1)$; then we conclude that

$$c \ge \frac{c}{\delta} + \frac{c\alpha}{\delta} + \alpha \iff c\left(1 - \frac{1}{\delta} - \frac{\alpha}{\delta}\right) = c\frac{\delta-\alpha-1}{\delta} \ge \alpha \iff c \ge \frac{\alpha\delta}{\delta-\alpha-1} \,.$$ □

[2] We could allow Steiner nodes and use a Steiner tree approximation, but this would not improve the worst-case bound.

We note that Lemma 4.3 explains how we circumvent a major obstacle to an $O(1)$-simultaneous approximation—the $\Omega(\log n)$ distortion lower bound for embedding arbitrary metrics into tree metrics [4]. If we needed to maintain distances between many pairs of nodes the task would be hopeless, but Lemma 4.3 shows that it suffices to preserve the distance of each node to the next layer, so the graph of distances to be maintained forms a tree.

We can now complete the proof of our main theorem and choose the optimal parameters.

Theorem 4.4. *The tree T achieves a simultaneous approximation ratio of $(1+\varepsilon)\lambda(8+4\sqrt{5})$ using a λ-approximation to SSRoB. In particular,*

- *there is a randomized polynomial time algorithm that finds a 47.45 simultaneous approximation with high probability,*
- *there is a deterministic polynomial time algorithm that finds a 55.58 simultaneous approximation, and*
- *there exists a tree that is a 16.95 simultaneous approximation.*

Proof. Applying Lemma 3.2 with $c_B = \beta\gamma/(\gamma-1)$ (Lemma 4.2) and $c_R = \alpha\delta/(\delta-\alpha-1)$ (Lemma 4.3), the final approximation ratio for an arbitrary cost function f is

$$(1+\varepsilon)\lambda \max\left\{\frac{\beta\gamma^2}{\gamma-1}, \frac{\alpha\delta^2}{\delta-\alpha-1}\right\}. \tag{4.1}$$

where the extra $1+\varepsilon$ comes from the approximation of f by a combination of A_i's. Now it is a simple matter of applying calculus to find the optimal values for α, β, γ, and δ.

The easiest parameters to fix are γ and δ. For γ:

$$\frac{d}{d\gamma}\left[\frac{\beta\gamma^2}{\gamma-1}\right] = \beta\frac{(2\gamma)(\gamma-1)-\gamma^2}{(\gamma-1)^2} = 0 \quad\Longrightarrow\quad \gamma(\gamma-2)=0 \quad\Longrightarrow\quad \gamma=2\,.$$

For δ:

$$\begin{aligned}\frac{d}{d\delta}\left[\frac{\alpha\delta^2}{\delta-\alpha-1}\right] = \alpha\frac{2\delta(\delta-\alpha-1)-\delta^2}{(\delta-\alpha-1)^2} = 0 \quad&\Longrightarrow\quad \delta^2-2\alpha\delta-2\delta = \delta(\delta-2\alpha-2)=0\\ &\Longrightarrow\quad \delta = 2\alpha+2\,.\end{aligned}$$

Plugging $\gamma = 2$ and $\delta = 2\alpha+2$ into (4.1), $\beta\gamma^2/(\gamma-1) = 4\beta$ and

$$\frac{\alpha\delta^2}{\delta-\alpha-1} = \frac{\alpha(2\alpha+2)^2}{(2\alpha+2)-\alpha-1} = 4\alpha(\alpha+1)$$

so (4.1) is now $\max\{4\beta, 4\alpha(\alpha+1)\}$. The constraints on α and β require $\beta \geq (\alpha+1)/(\alpha-1)$ [25], so one term blows up if the other shrinks. To minimize the maximum set the two expressions to be equal:

$$\beta = \frac{\alpha+1}{\alpha-1} = \alpha(\alpha+1) \quad\Longrightarrow\quad \alpha(\alpha-1)=1 \quad\Longrightarrow\quad \alpha^2-\alpha-1=0 \Longrightarrow \alpha = \frac{1\pm\sqrt{5}}{2}\,.$$

Using $\alpha = \frac{1+\sqrt{5}}{2}$, we get

$$\beta = \frac{\alpha+1}{\alpha-1} = \frac{3+\sqrt{5}}{-1+\sqrt{5}} = \frac{(3+\sqrt{5})(1+\sqrt{5})}{4} = \frac{8+4\sqrt{5}}{4} = 2+\sqrt{5}$$

and $\delta = 2\alpha+2 = 3+\sqrt{5}$.

Summarizing, we set

$$\alpha = \frac{1+\sqrt{5}}{2}, \qquad \beta = 2+\sqrt{5}, \qquad \gamma = 2, \qquad \text{and} \qquad \delta = 3+\sqrt{5},$$

for which

$$\frac{\beta\gamma^2}{\gamma-1} = \frac{\alpha\delta^2}{\delta-\alpha-1} = 4(2+\sqrt{5}),$$

so the simultaneous approximation ratio is $(1+\varepsilon)\lambda(8+4\sqrt{5})$. The final ratio depends on the type of algorithm desired:

- Using the best randomized approximation $\lambda = 2.8$, and the ratio is 47.45 with high probability.
- Using the best deterministic approximation $\lambda = 3.28$, and the ratio is 55.58.
- If the algorithm is allowed to run in exponential time $\lambda = 1$, and the ratio is 16.95. □

The 2.8-approximation of Eisenbrand et al. [7] actually provides a slightly stronger guarantee on R_i and B_i, and we can use Corollary 3.3 to get a tiny improvement in the approximation ratio at the cost of a more complex derivation:

Theorem 4.5. *There is a randomized polynomial time algorithm that finds a 47.07 simultaneous approximation with high probability.*

Proof. Lemma 2 and Theorem 5 in [7] prove that

$$\mathbf{E}[R_i] \le 2R_i^* + \frac{.807}{x} M_i B_i^* \qquad \text{and} \qquad \mathbf{E}[M_i B_i] \le \rho(x+\varepsilon)R_i^* + \rho M_i B_i^*,$$

where $\rho = 1.39$ is the Steiner tree approximation ratio and $x \in (0,1]$ is a parameter. Applying Corollary 3.3 with

$$\mu_R = 2, \qquad \mu_B = \frac{.807}{x}, \qquad \nu_R = \rho(x+\varepsilon), \qquad \text{and} \qquad \nu_B = \rho,$$

the simultaneous ratio is bounded by

$$(1+\varepsilon)\max\left\{\frac{2\alpha\delta^2}{\delta-\alpha-1} + \frac{\rho(x+\varepsilon)\beta\gamma^2}{\gamma-1}, \frac{.807\alpha\delta^2}{x(\delta-\alpha-1)} + \frac{\rho\beta\gamma^2}{\gamma-1}\right\}. \tag{4.2}$$

For γ and δ, both expressions inside the max function are minimized exactly as above in Theorem 4.4 with $\gamma = 2$ and $\delta = 2\alpha + 2$. Also as in Theorem 4.4, β can be set to $\frac{\alpha+1}{\alpha-1}$, the minimum allowed by the constraints. Plugging in these values and simplifying, expression (4.2) becomes

$$4(\alpha+1)\max\left\{2\alpha + \frac{\rho(x+\varepsilon)}{\alpha-1}, \frac{.807\alpha}{x} + \frac{\rho}{\alpha-1}\right\}.$$

Set the two expressions inside the maximization to be equal, and solve the resulting quadratic equation in x to get (for $\varepsilon = 0$):

$$x = \frac{1}{2} - \frac{\alpha(\alpha-1)}{\rho} + \sqrt{\frac{\alpha^2(\alpha-1)^2}{\rho^2} - \frac{.193\alpha(\alpha-1)}{\rho} + \frac{1}{4}}$$

using that $x > 0$.

The problem is now to minimize

$$4(\alpha+1)\left(2\alpha + \frac{\rho x}{\alpha-1}\right) = 4\alpha(\alpha+1) + \frac{2\rho(\alpha+1)}{\alpha-1} + 4(\alpha+1)\sqrt{\alpha^2 - \frac{.193\rho\alpha}{\alpha-1} + \frac{\rho^2}{(\alpha-1)^2}}.$$

This expression is unwieldy to optimize analytically, but when $\rho = 1.39$ it achieves a minimum of about 47.068 for $\alpha \approx 1.5495$ (which makes $x \approx .5995$). For ε sufficiently small this gives a bound of 47.07 with high probability. □

We leave as an open question the problem of exploiting Corollary 3.3 to substantially improve the ratio.

4.1 Runtime

Let $t_{RoB}(n,m)$ be the running time of our SSRoB approximation on a graph with n vertices and m edges, which must be at least $\Omega(n)$ to write down the output. When ε is constant, running the SSRoB approximation for each i takes $O(t_{RoB}(n,m)\log D)$ time, or $O(t_{RoB}(n,m)\log D\log\log D)$ for a randomized algorithm. Subsequent loops in Algorithm 1 take $O(n\log D)$.

For each of the $O(\log D)$ iterations of Algorithm 2 we need to do a graph contraction and run the LAST algorithm, which requires computing the MST and shortest path trees, so the runtime is $O((m+n+t_{SP}(n,m)+t_{MST}(n,m))\log D)$, where $t_{SP}(n,m)$ and $t_{MST}(n,m)$ denote the runtimes of the shortest path and MST algorithms used. Using Dijkstra's algorithm, $t_{SP}(n,m) = O(m+n\log n)$ [11], which dominates the other steps in the loop. Combining the two algorithms, the total time is $O((t_{RoB}(n,m)+m+n\log n)\log D)$ for deterministic SSRoB algorithms and $O((t_{RoB}(n,m)\log\log D + m + n\log n)\log D)$ for randomized ones.

5 Remarks on the approximation ratio for SSRoB

The current-best algorithm for SSRoB [7] depends on the approximation ratio for Steiner tree, which has recently been reduced to 1.39 [6]. The improved SSRoB ratio does not currently appear in the literature, so we include the calculation for completeness. See the original paper for details.

Theorem 5.1 ([7, 6])**.** *There is a 2.8-approximation for SSRoB.*

Proof. By the proof of Theorem 6 in [7], there is an SSRoB algorithm with expected cost

$$\rho(MB^* + (x+\varepsilon)R^*) + 2R^* + 0.807\frac{MB^*}{x}$$

where ρ is the approximation ratio for Steiner tree and x is a parameter in $(0,1]$. Set the coefficients of R^* and MB^* to be equal and solve the resulting quadratic equation in x. For $\rho = 1.39$, choosing $x = .5735$ gives an approximation ratio of 2.80. □

6 Open problems

We have answered the open questions posed by Goel and Estrin [12] and Goel and Post [13], but there are several avenues for further work. Our simultaneous ratio of 47.45 already improves upon many algorithms for normal SSBaB and is only a factor of 2.33 away from the best. It would be nice to eliminate this gap or, alternately, prove that a gap exists between the approximation achievable for fixed f and the best simultaneous ratio. We know of no lower bounds on what simultaneous ratio may be possible, so any progress in this direction would also be interesting. Generalizing the settings in which $O(1)$ simultaneous ratios are possible would be interesting, but may be unlikely given that one must contend with lower bounds for metric tree embedding [4] and multi-sink buy-at-bulk [1].

Acknowledgements

We thank the anonymous conference and journal reviewers for many helpful comments that improved the presentation and for suggesting Corollary 3.3 and Theorem 4.5.

References

[1] MATTHEW ANDREWS: Hardness of buy-at-bulk network design. In *Proc. 45th FOCS*, pp. 115–124. IEEE Comp. Soc. Press, 2004. [doi:10.1109/FOCS.2004.32] 364

[2] MATTHEW ANDREWS AND LISA ZHANG: The access network design problem. In *Proc. 39th FOCS*, pp. 40–49. IEEE Comp. Soc. Press, 1998. [doi:10.1109/SFCS.1998.743427] 352

[3] BARUCH AWERBUCH AND YOSSI AZAR: Buy-at-bulk network design. In *Proc. 38th FOCS*, pp. 542–547. IEEE Comp. Soc. Press, 1997. [doi:10.1109/SFCS.1997.646143] 354

[4] YAIR BARTAL: Probabilistic approximation of metric spaces and its algorithmic applications. In *Proc. 37th FOCS*, pp. 184–193. IEEE Comp. Soc. Press, 1996. [doi:10.1109/SFCS.1996.548477] 354, 361, 364

[5] YAIR BARTAL: On approximating arbitrary metrics by tree metrics. In *Proc. 30th STOC*, pp. 161–168. ACM Press, 1998. [doi:10.1145/276698.276725] 354

[6] JAROSLAW BYRKA, FABRIZIO GRANDONI, THOMAS ROTHVOSS, AND LAURA SANITÀ: An improved LP-based approximation for Steiner tree. In *Proc. 42nd STOC*, pp. 583–592. ACM Press, 2010. [doi:10.1145/1806689.1806769] 354, 355, 363, 364

[7] FRIEDRICH EISENBRAND, FABRIZIO GRANDONI, THOMAS ROTHVOSS, AND GUIDO SCHÄFER: Connected facility location via random facility sampling and core detouring. *J. Comput. System Sci.*, 76(8):709–726, 2010. [doi:10.1016/j.jcss.2010.02.001] 352, 354, 355, 358, 362, 363, 364

[8] MIHAELA ENACHESCU, ASHISH GOEL, RAMESH GOVINDAN, AND RAJEEV MOTWANI: Scale-free aggregation in sensor networks. *Theoret. Comput. Sci.*, 344(1):15–29, 2005. Preliminary version in ALGOSENSORS'04. [doi:10.1016/j.tcs.2005.06.023] 354

[9] MATTHIAS ENGLERT AND HARALD RÄCKE: Oblivious routing for the L_p-norm. In *Proc. 50th FOCS*, pp. 32–40. IEEE Comp. Soc. Press, 2009. [doi:10.1109/FOCS.2009.52] 354

[10] JITTAT FAKCHAROENPHOL, SATISH RAO, AND KUNAL TALWAR: A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. System Sci.*, 69(3):485–497, 2004. Preliminary version in STOC'03. [doi:10.1016/j.jcss.2004.04.011] 353, 354

[11] MICHAEL L. FREDMAN AND ROBERT ENDRE TARJAN: Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34:596–615, 1987. Preliminary version in FOCS'84. [doi:10.1145/28869.28874] 363

[12] ASHISH GOEL AND DEBORAH ESTRIN: Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk. *Algorithmica*, 43(1):5–15, 2005. Preliminary version in SODA'03. [doi:10.1007/s00453-005-1155-0] 353, 354, 364

[13] ASHISH GOEL AND IAN POST: An oblivious O(1)-approximation for single source buy-at-bulk. In *Proc. 50th FOCS*, pp. 442–450. IEEE Comp. Soc. Press, 2009. [doi:10.1109/FOCS.2009.41] 353, 364

[14] ASHISH GOEL AND IAN POST: One tree suffices: A simultaneous O(1)-approximation for single-sink buy-at-bulk. In *Proc. 51st FOCS*, pp. 593–600. IEEE Comp. Soc. Press, 2010. [doi:10.1109/FOCS.2010.62] 353

[15] FABRIZIO GRANDONI AND GIUSEPPE F. ITALIANO: Improved approximation for single-sink buy-at-bulk. In *17th Int. Symp. on Algorithms and Computation (ISAAC'06)*, volume 4288, pp. 111–120. Springer, 2006. [doi:10.1007/11940128_13] 354

[16] FABRIZIO GRANDONI AND THOMAS ROTHVOSS: Network design via core detouring for problems without a core. In *Proc. 37th Internat. Colloq. on Automata, Languages and Programming (ICALP'10)*, pp. 490–502. Springer, 2010. [doi:10.1007/978-3-642-14165-2_42] 352, 353, 354

[17] SUDIPTO GUHA, ADAM MEYERSON, AND KAMESH MUNAGALA: A constant factor approximation for the single sink edge installation problems. In *Proc. 33rd STOC*, pp. 383–388. ACM Press, 2001. [doi:10.1145/380752.380827] 352, 354, 356

[18] SUDIPTO GUHA, ADAM MEYERSON, AND KAMESH MUNAGALA: A constant factor approximation for the single sink edge installation problem. *SIAM J. Comput.*, 38(6):2426–2442, 2010. [doi:10.1137/050643635] 352, 354, 356

[19] ANUPAM GUPTA, MOHAMMAD T. HAJIAGHAYI, AND HARALD RÄCKE: Oblivious network design. In *Proc. 17th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'06)*, pp. 970–979. ACM Press, 2006. [doi:10.1145/1109557.1109665] 354

[20] ANUPAM GUPTA, JON KLEINBERG, AMIT KUMAR, RAJEEV RASTOGI, AND BULENT YENER: Provisioning a virtual private network: A network design problem for multicommodity flow. In *Proc. 33rd STOC*, pp. 389–398. ACM Press, 2001. [doi:10.1145/380752.380830] 354

[21] ANUPAM GUPTA, AMIT KUMAR, MARTIN PÁL, AND TIM ROUGHGARDEN: Approximation via cost sharing: Simpler and better approximation algorithms for network design. *J. ACM*, 54(3):1–38, 2007. [doi:10.1145/1236457.1236458] 354, 356

[22] ANUPAM GUPTA, ARAVIND SRINIVASAN, AND ÉVA TARDOS: Cost-sharing mechanisms for network design. *Algorithmica*, 50(1):98–119, 2008. Preliminary version in APPROX'04. [doi:10.1007/s00453-007-9065-y] 354

[23] RAJA JOTHI AND BALAJI RAGHAVACHARI: Improved approximation algorithms for the single-sink buy-at-bulk network design problems. *J. Discrete Algorithms*, 7(2):249–255, 2009. Preliminary version in SWAT'04. [doi:10.1016/j.jda.2008.12.003] 354

[24] DAVID R. KARGER AND MARIA MINKOFF: Building Steiner trees with incomplete global knowledge. In *Proc. 41st FOCS*, pp. 613–623. IEEE Comp. Soc. Press, 2000. [doi:10.1109/SFCS.2000.892329] 352, 354

[25] S. KHULLER, B. RAGHAVACHARI, AND N. YOUNG: Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, 14(4):305–321, 1995. [doi:10.1007/BF01294129] 353, 354, 355, 356, 361

[26] BHASKAR KRISHNAMACHARI, DEBORAH ESTRIN, AND STEPHEN WICKER: Modelling data-centric routing in wireless sensor networks. Technical Report CENG 02-14, Univ. Southern Calif. Dept. EE - Systems (18 pp), 2002. PDF. 352

[27] Y. RABINOVICH AND R. RAZ: Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete Comput. Geom.*, 19(1):79–94, 1998. [doi:10.1007/PL00009336] 353

[28] R. RAVI AND F. S. SALMAN: Approximation algorithms for the traveling purchaser problem and its variants in network design. In *Proc. 7th Ann. European Symp. on Algorithms (ESA'99)*, pp. 696–696. Springer, 1999. [doi:10.1007/3-540-48481-7_4] 354

[29] F. S. SALMAN, J. CHERIYAN, R. RAVI, AND S. SUBRAMANIAN: Buy-at-bulk network design: Approximating the single-sink edge installation problem. In *Proc. 8th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA'97)*, pp. 619–628. ACM Press, 1997. [ACM:314397] 352, 354

[30] F. S. SALMAN, J. CHERIYAN, R. RAVI, AND S. SUBRAMANIAN: Approximating the single-sink link-installation problem in network design. *SIAM J. Optim.*, 11(3):595–610, 2001. [doi:10.1137/S1052623497321432] 352, 354

[31] CHAITANYA SWAMY AND AMIT KUMAR: Primal-dual algorithms for connected facility location problems. *Algorithmica*, 40(4):245–269, 2004. Preliminary version in APPROX'02. [doi:10.1007/s00453-004-1112-3] 354

[32] KUNAL TALWAR: The single-sink buy-at-bulk LP has constant integrality gap. In *Proc. 9th Ann. Conf. on Integer Programming and Combinatorial Optimization (IPCO '02)*, pp. 475–486. Springer, 2002. [doi:10.1007/3-540-47867-1_33] 354

[33] ANKE VAN ZUYLEN: Deterministic sampling algorithms for network design. *Algorithmica*, 60(1):110–151, 2011. Preliminary version in ESA'08. [doi:10.1007/s00453-009-9344-x] 354

[34] DAVID P. WILLIAMSON AND ANKE VAN ZUYLEN: A simpler and better derandomization of an approximation algorithm for single source rent-or-buy. *Oper. Res. Lett.*, 35(6):702–712, 2007. [doi:10.1016/j.orl.2007.02.005] 354, 355

AUTHORS

Ashish Goel
Associate Professor
Stanford University
ashishg@stanford.edu
http://www.stanford.edu/~ashishg/

Ian Post
graduate student
Stanford University
itp@stanford.edu

ABOUT THE AUTHORS

ASHISH GOEL is an Associate Professor of Management Science and Engineering and (by courtesy) Computer Science at Stanford University, and a member of Stanford's Institute for Computational and Mathematical Engineering. He received his Ph. D. in Computer Science from Stanford in 1999, advised by Serge Plotkin, and was an Assistant Professor of Computer Science at the University of Southern California from 1999 to 2002. His research interests lie in the design, analysis, and applications of algorithms; current application areas of interest include social networks, Internet commerce, and large scale data processing. Ashish is a recipient of an Alfred P. Sloan faculty fellowship (2004-06), a Terman faculty fellowship from Stanford, an NSF Career Award (2002-07), and a Rajeev Motwani mentorship award (2010). He was a co-author on the paper that won the best paper award at WWW 2009. Ashish currently holds the 3COM faculty fellowship in Stanford's School of Engineering, and serves on the technical advisory board of Twitter, Inc.

IAN POST is a graduate student at Stanford advised by Ashish Goel. His research interests are in algorithms.