Simultaneous Optimization for Concave Costs: Single Sink Aggregation or Single Source Buy-at-Bulk*

Ashish Goel †
Stanford University

Deborah Estrin ‡
University of California, Los Angeles

Abstract

We consider the problem of finding efficient trees to send information from k sources to a single sink in a network where information can be aggregated at intermediate nodes in the tree. Specifically, we assume that if information from j sources is traveling over a link, the total information that needs to be transmitted is $f(j)$. One natural and important (though not necessarily comprehensive) class of functions is those which are concave, non-decreasing, and satisfy $f(0) = 0$. Our goal is to find a tree which is a good approximation *simultaneously* to the optimum trees for *all* such functions. This problem is motivated by aggregation in sensor networks, as well as by buy-at-bulk network design.

We present a randomized tree construction algorithm that guarantees $\mathbf{E}[\max_f C_f/C^*(f)] \leq 1+\log k$, where C_f is a random variable denoting the cost of the tree for function f and $C^*(f)$ is the cost of the optimum tree for function f. To the best of our knowledge, this is the first result regarding simultaneous optimization for concave costs. We also show how to derandomize this result to obtain a deterministic algorithm that guarantees $\max_f C_f/C^*(f) = O(\log k)$. Both these results are much stronger than merely obtaining a guarantee on $\max_f E[C_f/C^*(f)]$. A guarantee on $\max_f E[C_f/C^*(f)]$ can be obtained using existing techniques, but this does not capture simultaneous optimization since no one tree is guaranteed to be a good approximation for all f simultaneously.

While our analysis is quite involved, the algorithm itself is very simple and may well find practical use. We also hope that our techniques will prove useful for other problems where one needs simultaneous optimization for concave costs.

1 Introduction

Consider the problem of constructing a tree to send information from several sources to a single sink in a given graph. One setting in which this problem arises is a wireless sensor network [15, 9] where several sensor nodes need to send sensed information to a centralized processing agent. In this setting, information can typically be *aggregated*, i.e., if information from two different sensors,

*A Preliminary version of this paper appeared in the proceedings of the fourteenth ACM-SIAM Symposium on Discrete Algorithms, 2003.

†Department of Management Science and Engineering and (by courtesy) Computer Science, Stanford University, Terman 311, Stanford CA 94305. Email: ashishg@stanford.edu. Research supported by NSF CAREER Award 0133968. This research was conducted when the author was at the University of Southern California.

‡Department of Computer Science and Laboratory for Embedded Collaborative Systems (LECS), University of California, Los Angeles. This work was supported by DARPA under grant DABT63-99-1-0011 as part of the SCAADS project. Email: destrin@cs.ucla.edu.

A and B, is routed through a third sensor, C, then C can aggregate the information from the two sources to achieve reduction in the number of bits that need to be transmitted. Constructing aggregation trees to minimize the total cost of transmitting information is a challenging problem, and is receiving significant attention from the sensor networks community [15, 14]; this problem is particularly important since the sensors have limited battery power, and wireless communication is very power intensive.

We can model aggregation by stipulating that if information from j sources is routed over a single link, the total information that needs to be transmitted is $f(j)$. The function f is called the "aggregation function". We will assume that f is concave and non-decreasing, and that $f(0) = 0$. The requirements $f(0) = 0$ and f being non-decreasing are natural; the requirement that f is concave is also natural and corresponds to the information theoretic requirement that the total information carried by j symmetric sources is a concave function of j. Functions which satisfy these requirements are called *canonical* aggregation functions in this paper. While this is an important class of functions, it is not necessarily comprehensive; some other interesting classes are mentioned in section 5.

The single source buy-at-bulk problem arises in a completely different setting but is identical, in abstraction, to the problem described above. In the single source buy-at-bulk problem, we need to design a network for sending information from a single source to several sinks. If a link supports k sinks then the cost of the link is proportional to $f(k)$, where f is concave, f(0) = 0, and f is non-decreasing. In the rest of this paper, we will use the terminology derived from the aggregation problem.

If the function f is fixed and known in advance then the problem is well understood. Instead, our goal is to construct a tree that is *simultaneously* good for all canonical aggregation functions f. The existence of such a tree is not immediately obvious, and it is surprising that a simple algorithm, which we present later, achieves a simultaneous logarithmic approximation for all canonical aggregation functions. Simultaneous optimization is important for this problem since the aggregation function is often not known in advance or is poorly understood; further there are settings where we need to construct an aggregation tree which would be used to route different types of sensed information, each with its own aggregation function.

Informally, our main result is a randomized algorithm that constructs an aggregation tree which provides a logarithmic approximation simultaneously for all canonical aggregation functions. This result can be derandomized using the method of conditional expectations. To the best of our knowledge, ours is the first result which addresses simultaneous optimization for concave costs; this is an exciting area in general and merits further consideration. Our results and related work are described after we define the problem formally.

Problem Definition:

Formally, we assume that we are given an undirected graph $G = (V, E)$ with n vertices and m edges. We also assume that we are given a set of of k sources, and a single sink t. All the sources and the sink must belong to V. Each edge $e \in E$ has a non-negative cost $c(e)$. An aggregation tree is a tree which contains the sink and all the sources, and may contain additional vertices from V. We will treat the sink as the root of the tree. Let $e = (u, v)$ be an edge in an aggregation tree, such that u is the parent of v in the tree. Then the demand routed through an edge is the number of sources in the subtree rooted at v.

We define $\mathcal{F}$ to be the class of all real-valued functions f defined on non-negative real numbers

such that f is concave, $f(0) = 0$, and f is non-decreasing. We call such functions "canonical" aggregation functions. If $d(e)$ is the demand routed through an edge e, then we will assume that the cost of using this edge is $c(e)f(d(e))$, where f is a canonical aggregation function. In general, there is no reason to believe that the optimum set of paths to send information from the sources to the sink must form a tree; however, it is easy to see that for canonical aggregation functions, there exists an optimum set of paths that form a tree, and hence, we can focus our attention on just finding the optimum tree.

As mentioned before, this problem has been well studied when f is known. In this paper we will study the problem of approximating the optimum aggregation tree for a canonical aggregation function f *without* knowing what f is. Given a deterministic algorithm D for constructing such a tree, let $C_D(f)$ denote the cost of the resulting aggregation tree. Also, let $C^*(f)$ denote the cost of the optimum aggregation tree for function f. We can define the approximation ratio $\mathcal{A}_D$ of D as $\max_{f \in \mathcal{F}} C_D(f)/C^*(f)$. Our goal now is to find an algorithm D that guarantees a small value for $\mathcal{A}_D$. We will refer to this deterministic problem as **Det**.

Given a randomized algorithm R for constructing an aggregation tree, let $C_R(f)$ be the random variable which denotes the cost of this tree for function f. For randomized algorithms, there are two natural ways of extending the deterministic problem **Det**:

Problem R1: Find a randomized algorithm that guarantees a small value for

$$\max_{f \in \mathcal{F}} \mathbf{E}[C_R(f)/C^*(f)], \text{ or}$$

Problem R2: Find a randomized algorithm that guarantees a small value for

$$\mathbf{E}[\max_{f \in \mathcal{F}} \{C_R(f)/C^*(f)\}].$$

Problem **R2** subsumes **R1**, is more interesting, and much harder: using Jensen's inequality and the convexity of max, it is easy to see that $\mathbf{E}[\max_f C_R(f)/C^*(f)] \geq \max_f \mathbf{E}[C_R(f)/C^*(f)]$ for any randomized algorithm R. Problem **R2** is an accurate abstraction of simultaneous optimization and is the main subject of study in this paper. Our algorithm for problem **R2** can be derandomized using the method of conditional expectations to obtain a deterministic guarantee for problem **Det**.

Related Work:

When f is known in advance, a sequence of interesting papers [3, 7, 2, 11, 17] led to a constant factor approximation for this problem by Guha, Meyerson, and Munagala [12] (the problem is known to be NP-Hard and MAX-SNP-Hard since it contains the Steiner tree problem as a special case). When f is known in advance, but can be different for different links, Meyerson, Munagala, and Plotkin [17] gave a randomized $O(\log k)$ approximation algorithm which was derandomized by Chekuri, Khanna, and Naor [8].

When f is not known, Awerbuch and Azar [3] demonstrated how the tree embeddings of Bartal [6] can be used to solve problem **R1**[1]. The tree-embedding step was subsequently improved by Bartal [7] to $O(\log n \log\log n)$, and more recently, by Fakcheroenphol, Rao, and Talwar [10] to $O(\log n)$, which is asymptotically optimal. As a consequence, the guarantee for problem **R1**

[1]They did not actually define problem **R1**; the result is implicit in their work.

also improves to $O(\log n)$. However, it is not clear how these techniques can be extended to give similar results for the more interesting problem **R2**. In fact, it is conceivable *a priori* that none of the trees produced by Bartal's algorithm (or that of Fakcheroenphol, Rao, and Talwar) would be simultaneously good for all canonical aggregation functions.

Two interesting special cases are

1. $f(x)$ is constant for $x \geq 1$, and
2. $f(x) = x$.

The first corresponds to finding good Steiner trees (multicast) and the second corresponds to finding shortest path trees (unicast). Khuller, Raghavachari, and Young [16] outlined an algorithm that (with minor modifications) results in an $O(1)$ approximation simultaneously for these two special functions. Awerbuch, Baratz, and Peleg independently obtained a slightly weaker guarantee for this problem in an unpublished manuscript [5], building on an earlier work for weight-diameter approximation [4]. Our results for problems **R2** and **Det** can be viewed as a generalization of the work of Khuller, Raghavachari, and Young, even though our techniques are quite unrelated.

Our Results:

We present a simple algorithm that achieves a guarantee of $1 + \log k$ for both problems **R1** and **R2**. The algorithm is outlined in section 2. The analysis is presented in section 3, which contains the main technical contributions of this paper. It is worth noting that there are no hidden constants in this result. The basic intuition is to first construct a solution with flows which are a power of two, and then show that instead of analyzing the cost of the tree for each canonical function, it is sufficient to compare the cost of the edges with flow 2^i to the optimal cost just for the function $f(x) = \min\{x, 2^i\}$. Invoking the probabilistic method [1], this immediately gives an existential guarantee of $1 + \log k$ (but not an algorithm) for problem **Det**.

We then use the method of conditional expectations combined with the constant factor approximation of Guha, Meyerson, and Munagala [12] (or a more recent one due to Talwar [20]) to derandomize our algorithm. This results in a polynomial time algorithm which provides an $O(\log k)$ guarantee for problem **Det**; the details are in section 4.

Our randomized algorithm is simple enough to be implemented in a realistic system if offline computation is permissible. In terms of research directions for sensor networks, this is a strong argument against trying to construct approximate models of the aggregation function f, since we can come up with a single tree that is approximately good for all such functions.

We believe that the techniques we develop in this paper will find use in other settings where we need to do simultaneous optimization for concave costs. We describe several future directions and open problems in section 5.

2 The Hierarchical Matching Algorithm

The hierarchical matching algorithm outlined below is essentially a simplification of the techniques presented by Meyerson, Munagala, and Plotkin [17] and Guha, Meyerson, and Munagala [12]; it is surprising that this simple algorithm has the strong properties outlined in the introduction and proved in section 3.

We will assume, without loss of generality, that the graph is complete and satisfies the triangle inequality; if not we will complete the graph using its shortest path metric. Assume for now that k is a power of two; this assumption can be easily removed as outlined in appendix A. Initially, set $T \leftarrow \emptyset$. When the algorithm terminates, T will be the set of edges in the final aggregation tree.

The hierarchical matching algorithm runs in $\log k$ phases. In each phase, we perform the following two steps:

1. *The Matching Step:* Find a min-cost perfect matching in the subgraph induced by S. Let (u_i, v_i) represent the i-th matched pair, where $1 \leq i \leq |S|/2$.

2. *The Random Selection Step:* For all matched pairs (u_i, v_i), choose one out of u_i and v_i with probability half, and remove it from S.

In each phase, the size of S gets halved. After $\log k$ phases, $|S| = 1$. The algorithm then outputs the union of each of the $\log k$ matchings, and also outputs the edge connecting the single remaining element in S to the sink t. The set of output edges is the aggregation tree produced by the algorithm.

3 Analysis

In section 3.1, we will prove some useful lemmas and give a simple proof that the hierarchical matching algorithm guarantees $\mathbf{E}[C(f)/C^*(f)] \leq 1 + \log k$. This is a slight improvement over the $O(\log n)$ guarantee that can be obtained using tree embeddings [3, 6, 7, 10] for problem **R1**, but is not our main result.

Section 3.2 proves the main result of this paper, i.e., the hierarchical matching algorithm guarantees

$$\mathbf{E}\left[\max_{f \in \mathcal{F}} \left\{ \frac{C(f)}{C^*(f)} \right\}\right] \leq 1 + \log k.$$

As pointed out in the introduction, this is a much stronger statement since it allows us to construct a single aggregation tree that is *simultaneously* good for all canonical aggregation functions. There is no previously known method for obtaining such results via probabilistic tree embeddings[2].

3.1 Preliminaries

Let S_i denote the set of source vertices which still belong to S at the end of the i-th phase; we will use S_0 to denote the original set of sources. Let X_i denote the set of edges in the matching found in phase i of the algorithm, for $1 \leq i \leq \log k$. Also, let $X_{1+\log k}$ denote the edge connecting the vertex in the singleton set $S_{\log k}$ to the sink t. An edge in X_i carries aggregated data from 2^{i-1} sources. For any concave function f, define M_i to be the quantity $\sum_{e \in X_i} c(e)$. $M_i(f)$ represents the cost of the matching found in the i-th step. Clearly, $\sum_{i=1}^{1+\log k} M_i \cdot f(2^{i-1})$ is the cost of the tree T for aggregation function f. Further, let $C_i^*(f)$ denote the cost of the optimum aggregation problem where S_i is the set of sources, and each source wants to transmit 2^i units of data. Since the set of vertices to be deleted is chosen at random in each phase, $M_i(f)$ and $C_i^*(f)$ are all random variables.

[2] In hindsight, the techniques we develop in this paper can be used to prove that the tree embeddings of Fakcheroenphol, Rao, and Talwar [10] give a guarantee of $O(\log n \log k)$ for problem **R2**. This is much weaker that the $1 + \log k$ guarantee that our hierarchical matching algorithm provides, and hence, the details are omitted.

Lemma 3.1 *The sequence* $\langle C_0^*(f), C_1^*(f), C_2^*(f), \ldots, C_{\log k}^*(f)\rangle$ *is a super-martingale, i.e.,*

$$\mathbf{E}[C_i^*(f)|C_{i-1}^*(f)] \leq C_{i-1}^*(f).$$

Proof: For $0 \leq j \leq k/2^{i+1}$, let $(u_{i,j}, v_{i,j})$ represent the j-th pair in the matching constructed in the i-th phase of the hierarchical matching algorithm. Rather than prove the lemma directly for the sequence $C_i^*(f)$, we will define and analyze a super-sequence $D_{i,j}$. The super-sequence is defined for $0 \leq i \leq \log k$. For a given i, the value of j varies from 0 to $k/2^{i+1}$. The elements of the super-sequence are arranged in increasing order of i; elements with the same value of i are arranged in increasing order of j. Further:

1. $D_{i,0} = C_i^*(f)$.
2. For $0 \leq j \leq k/2^{i+1}$, $D_{i,j}$ is the cost of the optimum solution for the residual problem after j random selection steps during the i-th phase.

By definition, for $0 \leq i < \log k$, we have $D_{i,k/2^{i+1}} = C_{i+1}^*(f) = D_{i+1,0}$. In order to prove that the sequence $D_{i,j}$, and hence its sub-sequence $C_i^*(f)$, is a super-martingale, it suffices to show that the sequence $D_{i,j}$ is a super-martingale for a fixed value of i. Let $T_{i,j}$ denote the optimum tree for the residual problem after j random selection steps during the i-th phase, where $0 \leq j < k/2^{i+1}$. For an edge e in the tree $T_{i,j}$, let $d(e)$ denote the total demand routed through e. After the $(j+1)$-th selection step, let $d'(e)$ be the demand routed through this edge for the new residual problem, assuming that we continue to use tree $T_{i,j}$; note that the optimum tree for the new residual problem might be quite different. There are now three cases:

1. The edge e lies on the paths from the sink to both $u_{i,j+1}$ and $v_{i,j+1}$ in $T_{i,j}$
2. The edge e lies on neither of the paths
3. The edge e lies on one of the paths but not on the other

In cases 1 and 2 above, $d'(e) = d(e)$ regardless of which of the two vertices is chosen. In case 3, $d'(e) = d(e) + 2^i$ with probability 1/2 and $d'(e) = d(e) - 2^i$ with probability 1/2. Hence, in all cases, $\mathbf{E}[d'(e)] = d(e)$. We now apply Jensen's inequality [18]:

For any concave function f *and any random variable* X, $\mathbf{E}[f(X)] \leq f(\mathbf{E}[X])$.

Hence, $\mathbf{E}[f(d'(e)] \leq f(d(e))$. Summing over all edges in $T_{i,j}$, we can conclude that the expected cost of the tree $T_{i,j}$ for the residual problem after $j+1$ selection steps is no more than the cost of this tree for the residual problem after j selection steps. Using the fact that $T_{i,j}$ is the optimal tree for the residual problem after j selection steps, we obtain $\mathbf{E}[D_{i,j+1}] \leq D_{i,j}$, which completes the proof of this lemma. ■

Lemma 3.2 $M_i \cdot f(2^{i-1}) \leq C_{i-1}^*(f)$.

Proof: Let T_{i-1} denote the optimum tree for the residual problem after $i-1$ phases. Let $d(e)$ represent the amount of demand routed through edge e in tree T_{i-1}. Each surviving source has a demand of 2^{i-1}, and hence $d(e) \geq 2^{i-1}$ for any edge e in this tree. Now, $C_{i-1}^* = \sum_{e \in T_{i-1}} c(e) f(d(e))$. Since f is increasing, $C_{i-1}^* \geq f(2^{i-1}) \sum_{e \in T_{i-1}} c(e)$. An Eulerian tour of the tree T_{i-1} contains two disjoint matchings of all the sources, and hence $M_i \leq) \sum_{e \in T_{i-1}} c(e)$. Multiplying both sides of this inequality by $f(2^{i-1})$ gives us the desired result. ■

Theorem 3.3 *For any concave function f, $\mathbf{E}[C(f)/C^*(f)] \leq 1 + \log k$.*

Proof: For any concave function f,

$$\begin{aligned}
\mathbf{E}[C(f)] &= \mathbf{E}\left[\sum_{i=1}^{1+\log k} M_i \cdot f(2^{i-1})\right] \\
&= \sum_{i=1}^{1+\log k} \mathbf{E}[M_i \cdot f(2^{i-1})] \quad \text{[linearity of expectations]} \\
&\leq \sum_{i=1}^{1+\log k} \mathbf{E}[C^*_{i-1}(f)] \qquad \text{[using lemma 3.2]} \\
&\leq \sum_{i=1}^{1+\log k} C^*_0(f) \qquad \text{[using lemma 3.1]} \\
&= C^*(f)(1+\log k)
\end{aligned}$$

■

3.2 Hierarchical matching and problem R2

For $0 \leq i \leq k$, let A_i denote the following function from $\Re^+$ to $\Re^+$:

$$A_i(x) = \min\{x, 2^i\}.$$

We are going to call A_i the i-th "atomic" function. Clearly, A_i is a canonical aggregation function. Figure 1 illustrates this function pictorially. The basic intuition behind the rest of this section is that a good approximation of just the atomic functions results in a good approximation of all canonical aggregation functions.

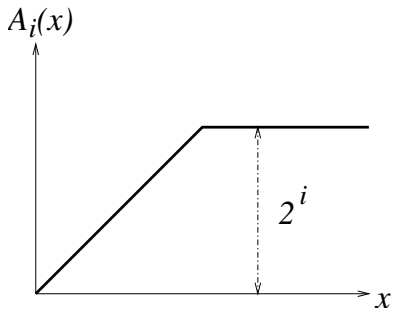

Figure 1: The i-th atomic function $A_i(x)$.

Let Γ denote the quantity $\sum_{i=1}^{\log k} M_i \cdot 2^{i-1}/C^*(A_{i-1})$. Note that Γ is a random variable which depends on the choices made during the hierarchical matching algorithm. The following series of lemmas are the main technical lemmas of this paper. The first of these illustrates a connection between atomic functions and arbitrary canonical functions.

Lemma 3.4 $C^*(f) \geq f(2^i)C^*(A_i)/2^i$.

Proof: Let $p(e)$ denote the number of sources that use edge e to communicate to the sink in the optimum aggregation tree. Clearly, $C^*(f) = \sum_e c(e)f(p(e))$, We define $p'(e) = \min\{p(e), 2^i\}$. Since f is increasing, $C^*(f) \geq \sum_e c(e)f(p'(e))$. Since $0 \leq p'(e) \leq 2^i$, we can think of $p'(e)$ as the convex combination of the numbers 0 and 2^i. Specifically, $p'(e) = 2^i \cdot (p'(e)/2^i) + 0 \cdot (1 - p'(e)/2^i)$. Invoking the concavity of f, we obtain $f(p'(e) \geq (p'(e)/2^i) \cdot f(2^i) + (1 - p'(e)/2^i) \cdot f(0)$. Using the fact that $f(0) = 0$, we can conclude that $f(p'(e)) \geq (p'(e)/2^i) \cdot f(2^i)$. Consequently, $C^*(f) \geq \sum_e (f(2^i)c(e)p'(e))/2^i$, which can be simplified to $C^*(f) \geq (f(2^i)/2^i) \sum_e (c(e)p'(e))$. Observe that $p'(e)$ is in fact exactly $A_i(p(e))$. Since $C^*(A_i)$ is the cost of the optimum tree for the function A_i, we have $\sum_e (c(e)p'(e)) \geq C^*(A_i)$, which completes the proof of the lemma. ■

Lemma 3.5 *For any canonical aggregation function f, $C(f)/C^*(f) \leq \Gamma$.*

Proof: Recall that $C(f) = \sum_{i=1}^{1+\log k} M_i \cdot f(2^{i-1})$. Therefore, $C(f)/C^*(f) = \sum_{i=1}^{1+\log k} M_i \cdot f(2^{i-1})/C^*(f)$. From lemma 3.4, we obtain $f(2^{i-1})/C^*(f) \leq 2^{i-1}/C^*(A_{i-1})$, which implies $C(f)/C^*(f) \leq \sum_{i=1}^{1+\log k} M_i \cdot 2^{i-1}/C^*(A_{i-1}) = \Gamma$. ■

The next lemma places an upper bound on the expected value of the quantity $M_i \cdot 2^{i-1}/C^*(A_{i-1})$.

Lemma 3.6 $\mathbf{E}[M_i \cdot 2^{i-1}/C^*(A_{i-1})] \leq 1$.

Proof: Recall that $C^*_j(f)$ is the cost of the optimal solution to the residual problem after j iterations of the hierarchical matching algorithm. In the residual problem, there are $k/2^j$ remaining sources, and each source has a "demand" of 2^j. We are going to consider the quantity $C^*_{i-1}(A_{i-1})$. From lemma 3.2, $M_i \cdot A_{i-1}(2^{i-1}) \leq C^*_{i-1}(A_{i-1})$. From lemma 3.1, $\mathbf{E}[C^*_{i-1}(A_{i-1})] \leq C^*(A_{i-1})$. Combining both lemmas, we obtain $\mathbf{E}[M_i] \cdot A_{i-1}(2^{i-1}) \leq C^*(A_{i-1})$. Since $A_{i-1}(2^{i-1}) = 2^{i-1}$, we obtain the desired result. ■

It is now straight-forward to obtain the main result of this paper:

Theorem 3.7 *For the hierarchical matching algorithm, $\mathbf{E}[\max_{f \in \mathcal{F}} C(f)/C^*(f)] \leq 1 + \log k$.*

Proof: An equivalent statement of lemma 3.5 is that $\max_{f \in \mathcal{F}} C_f/C^*(f) \leq \Gamma$, and hence,

$$\mathbf{E}[\max_{f \in \mathcal{F}} C_f/C^*(f)] \leq \mathbf{E}[\Gamma].$$

By definition of Γ,

$$\mathbf{E}[\Gamma] = \sum_{i=1}^{\log k} \mathbf{E}[M_i] \cdot 2^{i-1}/C^*(A_{i-1}).$$

The theorem now follows by applying lemma 3.6. ■

Thus the hierarchical matching algorithm gives a guarantee of $1 + \log k$ for problem **R2**.

4 Derandomization

The basic idea is to use the method of conditional expectations, combined with pessimistic estimators [19, 18]. We will first find trees $T_0, T_1, \ldots, T_{\log k}$ such that T_i is an α-approximation to the optimum tree for function A_i. Formally, $C_{T_i}(A_i) \leq \alpha C^*(A_i)$, where $C_T(f)$ represents the cost of tree T for aggregation function f. We can find such trees for $\alpha = O(1)$ using the constant factor approximation algorithm of Guha, Meyerson, and Munagala [12] for the single commodity buy-at-bulk problem with a fixed concave function, or a more recent approximation algorithm due to

Talwar [20]). Since the functions A_i have a linear-increase region followed by a constant region, we can also use an approximation algorithm by Gupta *et al.* [13] for the multicommodity rent-or-buy problem to obtain smaller constants ($\alpha = 12$).

Suppose we could systematically derandomize the hierarchical matching algorithm to guarantee that

$$\sum_{i=1}^{1+\log k} M_i 2^{i-1} / C_{T_{i-1}}(A_{i-1}) \leq 1 + \log k. \tag{1}$$

Since $C_{T_i}(A_i) \leq \alpha C^*(A_i)$, and $\alpha = O(1)$, condition 1 would guarantee that $\Gamma = O(\log k)$, proving the result we need. Hence we will now focus on ensuring that condition 1 is satisfied. Informally, the idea is to use the trees $T_0, T_1, \ldots, T_{\log k}$ as reference points for a pessimistic estimator.

Consider the j-th selection step of the i-th phase, where i goes from 1 to $1 + \log k$. Matchings $M_1, M_2, \ldots, M_{i-1}$ have already been decided. Let $S_{i,j}$ denote the set of remaining sources, and let $d_{i,j}(x)$ denote the demand at a node $x \in S_{i,j}$. Note that $d_{i,j}(x)$ must be either 2^i or 2^{i-1}. Let $c_{i,j}(q)$ denote the cost of tree T_q for satisfying demands $d_{i,j}$, assuming that the aggregation function is A_q. We define an estimator $E_{i,j}$ as follows:

$$E_{i,j} = \sum_{q=1}^{i-1} M_q 2^{q-1} / C_{T_{q-1}}(A_{q-1}) + \sum_{q=i}^{1+\log k} c_{i,j}(q-1) / C_{T_{q-1}}(A_{q-1}).$$

Clearly, $E_{1,0} = \sum_{q=1}^{1+\log k} c_{1,0}(q-1)/C_{T_{q-1}}(A_{q-1}) = \sum_{q=1}^{1+\log k} C_{T_{q-1}}(A_{q-1})/C_{T_{q-1}}(A_{q-1}) = 1 + \log k$. Also, the final value of the estimator is $E_{1+\log k,0} = \sum_{i=1}^{1+\log k} M_i 2^{i-1}/C_{T_{i-1}}(A_{i-1})$. If we could present an algorithm that would ensure that the estimator never increases, we would have ensured condition 1.

During the matching step, the same argument as in lemma 3.2 ensures that the estimator can not increase. During a random selection step, the same argument as lemma 3.1 guarantees that the *expected* new value of the estimator after the random selection is no larger than the value before the selection. But there are just two choices during a random selection; we can try them both and pick one which does not result in an increase in the value of the estimator. This is now a *deterministic* selection step, and hence we have an $O(\log k)$ guarantee for problem **Det**.

Admittedly, the derandomization is only of technical interest. It lacks the clean simplicity of the randomized algorithm and results in a much worse performance guarantee in terms of constant factors. For practical applications we would recommend using the randomized algorithm, specially since problem **R2** already captures simultaneous optimization quite well.

5 Open Problems

We studied simultaneous optimization for concave costs in the context of constructing good aggregation trees. It is our hope that the techniques and the framework developed in this paper will find more widespread use. One obvious open problem is to find matching upper and lower bounds on the best guarantee for problem **R2**. Our conjecture is that the best guarantee possible would be super-constant. Some other directions are:

1. To study the problem when there can be multiple sources and multiple sinks, i.e., the multicommodity version of the problem studied here. Probabilistic tree embeddings [6, 7, 10] can be used to obtain a somewhat unsatisfying $O(\log n \log k)$ guarantee for problem **R2** in

this setting; the details are similar to the proofs in this paper and are omitted. Obtaining a logarithmic approximation for this problem remains an interesting open problem.

2. To study the problem where the amount of aggregation depends not just on the number of sources, but also on the identity of the sources. Before solving this problem, we need to develop computationally useful models of what constitutes a reasonable aggregation function in this setting. Again, this is an interesting problem even when f is fixed.

3. To study the problem where information can be consumed along the tree if an intermediate node realizes that the information is not useful. For example, a sensor in an orchard might sense a pest infestation, and send detailed information about this infestation up the aggregation tree. Another sensor might sense a slight increase in pesticides in the atmosphere, and send information about air quality up the tree. An intermediate node might be able to surmise that the slight increase in pesticide level is expected and desirable given the pest problem, and refrain from sending the two pieces of information up the tree. In this scenario, the aggregation function is not non-decreasing: $f(1) = 1$ but $f(2) = 0$. Again, adequate understanding and models of the problem are required before it can be tackled, and the problem is interesting even when f is fixed.

4. Suppose each source could output different amounts of information. For simplicity, assume that the output of each source is some integer, that the smallest output is 1 and the largest output is Δ. Our results can be used in a black-box fashion to give a $1 + \log k + \log \Delta$ guarantee for this new problem **R2**. It would be interesting to devise an algorithm that does not incur the additional $\log \Delta$ penalty.

References

[1] N. Alon and J. Spencer. *The Probabilistic Method (2nd Ed.)*. Wiley, 2000.

[2] M. Andrews and L. Zhang. The access network design problem. *Proceedings of 39th IEEE Symposium on Foundations of Computer Science*, 1998.

[3] B. Awerbuch and Y. Azar. Buy-at-bulk network design. *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 542–47, 1997.

[4] B. Awerbuch, A. Baratz, and D. Peleg. Cost-sensitive analysis of communication protocols. *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, pages 177–87, 1990.

[5] B. Awerbuch, A. Baratz, and D. Peleg. Efficient broadcast and light-weight spanners. *Manuscript*, 1991.

[6] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. *37th IEEE symposium on Foundations of Computer Science*, pages 184–193, 1996.

[7] Y. Bartal. On approximating arbitrary metrics by tree metrics. *30th ACM Symposium on Theory of Computing*, 1998.

[8] C. Chekuri, S. Khanna, and S. Naor. A deterministic algorithm for the cost-distance problem. *Proc. 12th Annual Symposium on Discrete Algorithms*, 2001.

[9] D. Estrin, R. Govindan, and J. Heidemann (Editors). Embedding the internet. *CACM special issue on embedding the Internet*, 43(5), May 2000.

[10] J. Fakcheroenphol, S. Rao, and K. talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Proceedings of ACM Sympoisum on Theory of Computing*, 2003.

[11] S. Guha, A. Meyerson, and K. Munagala. Hierarchical placement and network design problems. *Proceedings of 41st IEEE Symposium on Foundations of Computer Science*, 2000.

[12] S. Guha, A. Meyerson, and K. Munagala. A constant factor approximation for the single sink edge installation problem. *Proceedings of 33rd ACM Symposium on Theory of Computing*, 2001.

[13] A. Gupta, A. Kumar, M. Pal, and T. Roughgarden. Approximation via cost-sharing: A simple approximation algorithm for the multicommodity rent-or-buy problem. *Proceedings of 44th IEEE Symposium on Foundations of Computer Science*, pages 606–615, 2003.

[14] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. *To appear in Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, 2002.

[15] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. *ACM MobiCom*, 2000.

[16] S. Khuller, B. Raghavachari, and N. Young. Balancing minimum spanning and shortest path trees. *Algorithmica*, 14(4):305–321, 1994.

[17] A. Meyerson, K. Munagala, and S. Plotkin. Cost-distance: Two metric network design. *Proc of 41st IEEE Symposium on Foundations of Computer Science*, 2000.

[18] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[19] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer problems. *Journal of Comp. Sys. Sci.*, 37:130–43, 1988.

[20] K. Talwar. Single sink buy-at-bulk lp has constant integrality gap. *Integer Programming and Combinatorial Optimization*, 2002.

A Hierarchical matching when k is not a power of 2

If k is not a power of 2, add $2^{\lceil \log k \rceil} - k$ copies of the sink t to the set of sources S; these are called "fake" sources as distinct from the "original" sources. S now becomes a multiset. Clearly, the optimum solution for any canonical function f is the same for the new problem and the original problem, and a $1 + \lceil \log k \rceil$ guarantee is trivial to obtain. We can in fact do a little better. During any random selection step,

1. If both nodes (u, v) in the matched pair are original sources, then follow the random selection process outlined in section 2.

2. If both nodes (u, v) are fake sources, then pick one arbitrarily. The cost of matching two fake sources is zero, so it does not matter which one gets chosen.

3. If one of the nodes is a fake source, and the other is an original source, then discard the original source and choose the fake source as the leader. It is easy to see that this does not violate the Martingale property of lemma 3.1.

After $\lceil \log k \rceil$ steps, there is a single fake source remaining. This is in fact a copy of the sink t. The extra step in the hierarchical matching algorithm to connect this to the sink is unnecessary, and we obtain a guarantee of $\lceil \log k \rceil$ for problem **R2**. Since $\lceil \log k \rceil < 1 + \log k$, the $1 + \log k$ guarantee continues to hold.