# Distributed Resource Management and Matching in Sensor Networks

Jie Gao* Leonidas Guibas† Nikola Milosavljević† Dengpan Zhou*

*Department of Computer Science
Stony Brook University
Stony Brook, NY 11794
{jgao,dpzhou}@cs.sunysb.edu

†Department of Computer Science
Stanford University
Stanford, CA 94305
{guibas,nikolam}@cs.stanford.edu

## ABSTRACT

We consider a scenario in which there are *resources* at or near nodes of a network, which are either static (e.g. fire stations, parking spots) or mobile (e.g. police cars). Over time, *events* (fires, crime reports, cars looking for parking) arise one-by-one at arbitrary nodes, and need to be quickly matched to and serviced by an appropriate nearby resource, without knowledge of future requests, and without the ability to alter any decision once it has been made.

We develop distributed algorithms to direct available resources in the network to these events (or vice versa) in a coordinated fashion, so that no two resources are assigned to the same event, and the total distance of the events from their matched resources is minimized. The key idea is to extract, in a preprocessing stage, a *well-separated tree metric* that approximates the original network metric by a logarithmic distortion, allowing greedy matching algorithms to generate close to optimal matchings, and enabling communication-efficient probing-based algorithms for events to detect nearby available resources. The distributed matching algorithm requires no global coordination and achieves polylogarithmic performance ratio in both online and offline settings. Simulation experiments corroborate the theoretical results on solution quality and further evaluate the communication costs of our scheme in practice.

## Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network protocols—*Routing protocols*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

## General Terms

Algorithms, Design, Theory

## Keywords

Distributed Matching, Resource Management, Sensor Networks

## 1. INTRODUCTION

Recent advances in wireless sensor networks reveal the potential of such embedded networked systems for revolutionizing the way we observe, interact with, and influence the physical world. Applications of sensor networks now extend beyond military deployments and the monitoring of animal or other natural habitats to places where humans work and live: homes, cars, buildings, roads, cities, etc. In these human spaces a sensor network serves users embedded in the same physical space as the network, allowing real-time data acquisition, situational understanding, event response and control, eventually leading to a fully intelligent living environment.

In this paper we explore the challenge of using a network of embedded sensors in aiding information discovery and resource management so as to allow coordinated response to distributed emerging events. The embedded sensors serve two purposes: discovering/detecting the events of interest (e.g., a parking spot is left empty); and forming a supporting infrastructure for distributed resources/users to act on the detected events (e.g., help a vehicle to look for an empty parking spot). In the parking scenario, multiple vehicles can be consulting with the sensor network for available parking spots at the same time. The sensor network needs to resolve competition and match vehicles with empty parking spots, in a way that avoids directing two cars to the same empty parking spot. In another scenario, emergency events detected by sensor nodes such as abnormalities, sensor battery outages, or local data overflows need to be handled immediately. Thus the sensor network faces the problem of directing surveillance vehicles/network helpers/data mules, one per event, to support real-time response and maintain network normal functioning.

When multiple events arise with multiple available resources to possibly act on them, there is an immediate need for coordinated and distributed resource management, to assign the resources to these events in a balanced yet efficent way. Here the *event* refers to, for example, a new vehicle querying for parking spots, or an emergency situation waiting for rescue efforts; and the *resource* refers to available parking spots or rescue teams. We model the resource management problem as follows: there are $k$ available resources residing in the network, events may emerge at any time near any node, and multiple events can appear simultaneously. Both available resources and these emerging events are detected and tracked by their nearby sensor nodes. We would like to design a distributed matching mechanism to assign each event to a distinct available resource. Naturally, to reduce response delay and energy expense, an event should be matched to a nearby resource, with the distance measured either in the network metric (the number of hops in the network from the resource location to the event location) or other underlying metric (e.g., the Euclidean distance metric), depending on how the event is to be serviced. The quality of the solution is naturally represented by the total distance of the matching.

This resource management problem, in the centralized setting, is simply a classical minimum cost bipartite matching problem. Each resource $i$ is connected to each event $j$ with an edge of cost equal to the distance between them (in the appropriate sense). The optimal matching with the minimum total cost of the edges in the matching can be found by flow algorithms in $O(k^3)$ running time for $k$ events and $k$ resources [16]. However, implementation of the centralized min-cost matching solution is not feasible in the sensor network setting for two reasons. First this would require the collection of all the information on the resources and events at a central place to execute the flow algorithm, which we would like to avoid. In addition, the events may emerge anywhere anytime and often require immediate response. Thus we deal with an online setting in which resource commitments must be made before we know the entire event sequence in the future. How to achieve the coordination needed in the matching algorithm in a distributed setting, with the events coming online, not aware of each other and not aware of the locations of available resources, is the main problem we will tackle in this paper.

## 2. OVERVIEW

For distributed resource management, we need to solve the following two problems:

- low-cost resource discovery and aggregation: when an event emerges, how does it discover nearby available resources in a communication efficient way; and
- close to optimal matching algorithm: which resource should an event select to be matched with, and how to resolve competition, so that no two events select the same resource.

The two problems, *resource discovery* and *distributed matching*, are closely related. The solution for one may impact the solution for the other.

Let us first consider a typical setting when a set of events pop up at the same time. In a distributed setting where no sensor node is taking full control, the challenge for the resource discovery problem is to decide what information is to be sent, and to where. The naive solution of flooding the entire network with all the events allows each event to solve for the centralized min-cost matching solution, but is obviously too communication expensive. To reduce communication cost, one may use restricted flooding to discover the closest available resource from each event. In particular, the TTL of the flooding messages from an event doubles until a resource is found. The communication cost, measured in the number of message transmissions, is $O(\ell^2)$, if the closest resource is $\ell$ hops away. In this way, each event discovers the closest resource that has not yet been matched with others. The problem is that such a greedy matching algorithm may give a very poor approximation ratio of $\Theta(k^{\lg_2 3})$ to the min-cost matching of $k$ events [17]. To get a better approximation ratio, one may adopt the 2-approximation algorithm for min-cost matching by the primal-dual method [12]. This would require a flooding from each event and global coordination of the growth of each aggregated component, thus incurring a much higher total communication cost.

From the above discussion, it is clear that we need to find a solution that achieves a balance between a low communication cost for resource discovery and a good approximation ratio for distributed matching. Our solution is to define a tree metric from the underlying network metric (minimum hop count distance metric of the sensor network) so that

- the simple greedy algorithm in the tree metric gives a polylogarithmic approximation ratio (or competitive ratio in the online setting when the events appear one by one [7]);
- the tree structure allows easy probing based detection of nearby resources, thus significantly saving communication cost (when compared with flooding the network).

The tree metric we extract is based on a *hierarchical well-separated tree*. An $\alpha$-hierarchically well-separated tree ($\alpha$-HST) is defined as a rooted weighted tree such that: the weight of all edges between a node and its children are the same; the edge weights along any path from the root to a leaf are decreasing by a factor of $\alpha$. The nodes of the HST are the nodes in the original communication network $G$, but a tree edge is virtual, i.e., it maps to a path in the original network.

This hierarchical well-separated tree is going to approximate the original graph metric $G$ by a logarithmic distortion factor, in terms of the shortest path distance between any two nodes. In particular, what we propose to extract is one $\alpha$-HST, chosen by a distribution $\mathcal{D}$ from a family $\mathcal{S}$ of $\alpha$-HSTs such that for each HST $T$ in the family $\mathcal{S}$, the distance $d_T(u, v)$ between any two nodes $u, v$ in the tree $T$ is greater than the shortest path length $d_G(u, v)$ in the original graph $G$, and that the expected distance between $u, v$, $E_{T \in \mathcal{S}}[d_T(u, v)D(T)]$, taken over the distribution $\mathcal{D}$ on the family $\mathcal{S}$ of trees, is no greater than $\beta \cdot d_G(u, v)$, where $D(T)$ is the probability of $T$ in the family $\mathcal{S}$. Such a family of HSTs is said to $\beta$-probabilistically approximate the original metric $G$, with $\beta$ being the (expected) distortion factor.

The hierarchical well-separated tree brings in two benefits to the distributed resource management problem. First, a HST is a special and simpler metric such that simple greedy algorithm by recursively matching closest pairs on the *tree*, instead of the original graph, gives good performance for both off-line and online matching. In the off-line (but distributed) setting when $k$ events appear at the same time, the greedy algorithm gives *optimal* min-cost matching on the HST. As the tree metric approximates the original graph metric by a logarithmic factor, this algorithm gives an $O(\log k)$ approximation ratio for the min-cost matching on the original communication graph. In the online setting when the events appear in any order and we compare the quality of the online matching with the off-line optimal solution (when the entire sequence of events is known), the simple greedy algorithm gives a competitive ratio of $O(\log^2 k)$ compared with the optimal matching on the HST, and consequently $O(\log^3 k)$ in the original metric. Secondly, working on the hierarchical well-separated tree solves information discovery and aggregation easily. We adopt a similar probing scheme as in our previous sparse aggregation framework [11]. Each event sends probes from along the HST looking for the nearest unmatched resource.

In section 4 we describe a distributed algorithm for the extraction of the probabilistic HST from the sensor network metric. The construction of HST is a preprocessing step that is executed during the network setup phase. Nodes od the HST correspond to nodes in the network; each network node maps to at least one tree node, possibly more. Edges of the HST correspond to paths in the network. Each such path is stored as a (bidirectional) chain of next-hop pointers, one per node along the path. Each HST node records information required to associate its outgoing paths to its adjacent HST edges, as well as the weights of those edges in the HST. In summary, to store the HST in the network, a network node has to store the amount of information proportional to the number of paths that go through it, and the number of HST edges adjacent to HST nodes that it represents.

For a particular resource management problem, we can condense the HST to get a tree with the leaf nodes only on the available $m$ resources with the total size of the tree being $O(m)$. For load balancing, we can build a few HSTs, with different random seeds, and rotate between them periodically. The construction of the HST assumes a random indexing of the nodes and proceeds in $O(\log n)$ rounds. At round $i$, only a subset of the nodes $P_i$ 'survive' to be qualified as the nodes on the $i$-th level of the tree (counted from the leaf level of the tree). These nodes flood the network with maximum hop count $\beta \cdot 2^i$, where $\beta$ is a number chosen randomly between $1/2$ and 1 in the initialization phase by a preconfigured 'leader' node, who bradcasts the value to the rest of the network. Every node selects, among all the messages it received at this round, the node with minimum index to be its ancestor at level $i + 1$ with an edge weight as $2^i$. The nodes that are not nominated by any node will quit after round $i$. Intuitively as $i$ increases, although the flooding radii increase, the number of candidates decreases. We show that the total communication cost for each round, in terms of the number of message transmissions, is bounded by $O(n)$, and the total communication cost for the construction of a HST is $O(n \log n)$, if the underlying communication graph has a constant doubling dimension[1].

The construction of the HST in a distributed environment is interesting on its own and can potentially have more applications besides the distributed matching algorithm for resource management explained above — working on a tree is much easier than working on the original graph; thus many optimization problems admit algorithms of improved performance on a tree. The construction of the HST naturally extends the applicable domain of these algorithms to a sensor network setting with only a logarithmic factor loss.

## 3. PREVIOUS WORK

The min-cost matching problem in a bipartite weighted graph can be solved optimally by the flow algorithm in a centralized setting. The greedy matching algorithm that matches closest pairs achieves a worst-case approximation ratio of $\Theta(k^{\lg_2 3})$ for $k$ vertices on a general graph or under Euclidean metric [17], but is optimal on a HST metric [3]. For the online min-cost matching problem, the $k$ vertices (blue vertices) on one side of the bipartite graph are given, the $k$ vertices (red vertices) on the other side of the graph are revealed one by one. When a red vertex appears, it needs to be matched with a blue vertex immediately. The solution is compared with the optimal matching when all the red vertices are given altogether and the performance degradation is

[1]The doubling dimension of a metric space $(X, d)$ is the smallest value $\rho$ such that each ball of radius $R$ can be covered by at most $2^\rho$ balls of radius $R/2$ [13]. If we place at most a constant number of sensor nodes inside any unit disk and the holes in the sensor networks are not very fragmenting, the communication graph has constant doubling dimension.

captured by the competitive ratio of the online decision making procedure. Meyerson *et al.* [15] proposed the first randomized online algorithm that achieves a logarithmic competitive ratio of $O(\log^3 k)$, by using the probabilistic HST approximation of a general graph metric. Essentially each red node is matched to the closest blue node on the HST and ties are broken randomly. An improved algorithm by Bansal *et al.* [3] achieves a competitive ratio of $O(\log^2 k)$.

Approximating a metric with probabilistic hierarchical well-separated trees was first proposed by Bartal [5, 4], with the motivation that many problems are easier to solve on a tree than on a general graph. Later, Fakcharoenphol *et al.* [9] improved the distortion to $O(\log n)$ for any $n$ node metric and this is tight.

This paper borrows from these ideas. This previous works, however, were for a centralized environment. In this paper the distributed algorithm design and the communication cost analysis for both (i) extracting a HST from the network metric and (ii) applying probing-based mechanisms on the HST for information discovery and resource management, in a resource constrained sensor network setting, are new.

The results in this paper is also related with our previous work on aggregation of sparsely located events, none of them with knowledge of each other [11]. The idea is to use lightweight local probes (in the vertical and horizontal directions) from the simultaneously emerging events to achieve distance sensitive neighbor discovery with nearby events discovering each other first. Aggregation is performed when multiple events 'find' each other and at the same time an aggregation tree is formed to suppress the probing of all but one event being aggregated. The probes that survive this aggregation will carry the aggregated information and propagate further to look for other possible events. The fact that aggregation is done naturally along a tree structure constructed by the individual probes substantially reduces the total communication cost for the group discovery of these events to only an $O(\log k)$ factor over the cost of the minimum spanning tree connecting all the events. The novelty in this paper is the different tree metric (the HST) that we use and the benefits of the special tree metric for the distributed matching problem. We show in Section 4.5 that the HST can also be used for sparse aggregation with the same communication cost.

## 4. HIERARCHICALLY WELL-SEPARATED TREES

In this section we give the definition of an $\alpha$-hierarchically well separated tree ($\alpha$-HST) and show how to compute the $\alpha$-HST embedding of a constant doubling dimension graph metric in a distributed way.

**Definition 4.1** (**$\alpha$-HST**). *A rooted weighted tree $T$ is an $\alpha$-HST if the weights of all edges between an internal node and its children are the same, all root-to-leaf paths have the same hop-distance, and the edge weights along any such path decrease by a factor of $\alpha$ as we go down the tree.*

Throughout this paper we 'count levels from leaves to the root', i.e., we define the *level* of a node $u$ in $\alpha$-HST as the distance *in hops* from $u$ to any of its leaves. In particular, all leaves have level 0. We define the level of a subtree to be the level of its root.

Fakcharoenphol *et al.* [9] give a centralized algorithm to compute, given an $n$-point metric $(P, d)$, a random 2-HST $T$ whose tree metric $d_T$ $O(\log n)$-*probabilistically* approximates $d$. That is, for any $u, v \in P$, $d_T(u, v) \geq d(u, v)$ and $E[d_T(u, v)] \leq O(\log n) \cdot d(u, v)$, where the expectation is taken over random choices of the algorithm, equivalently over the distribution on the resulting trees. In this paper, we use $P$ to denote the set of nodes in an undirected graph modeling the sensor network, and $d(u, v)$ denotes the minimum hop count between $u, v \in P$ in the sensor network. When we use a 2-HST to approximate the graph metric $d$, all the nodes in $P$ are placed as the leaf nodes of the 2-HST. The distance between two nodes $u, v \in P$ on the tree $T$, $d_T(u, v)$, is defined as the sum of the distances along the paths from $u, v$ to their lowest common ancestor in the 2-HST. The internal nodes in the HST are abstract nodes, although they might be labeled by some nodes of $P$, as will be explained later. We denote by $B(p, r)$ the ball centered at point $p$ with radius $r$.

The algorithm of [9] is centralized and executes in a top-down fashion to partition the nodes in the network hierarchically. The HST naturally corresponds to this hierarchical partition of $P$. The nodes of $P$ are the leaf nodes of the HST and each internal node in the HST corresponds to a cluster of nodes in the hierarchical partitioning. For completeness, we review this algorithm below. We fix a permutation $\pi : P \to \{1, 2, \ldots, n\}$ of the nodes, chosen uniformly at random from the set of all permutations. We also fix a value $\beta$ chosen uniformly at random from $[\frac{1}{2}, 1)$. To get a 2-HST, we compute for each node $u$ a $O(\log(n))$-dimensional *signature vector* $S(u)$, where the $i$-th element in the vector is

$$S(u)_i = \arg \min_{v \in B(u, 2^i \beta)} \pi(v) , \qquad (1)$$

for $i = 0$ to $m = \lceil \log D \rceil + 1$ with $D$ the network diameter. That is, each node keeps the node with the smallest rank among all nodes within distance $2^i \beta$. For all nodes $u$, $S(u)_m$ is the node with rank 1. These signature vectors define the HST embedding of $d$. In particular, the leaves are nodes of $P$, the level-$i$ ancestor of a node $u$ is labeled $S(u)_i$, and the weights of all edges between level $i$ and level $i - 1$ is $2^i$. The fact that this is indeed a 2-HST (in the sense of Definition 4.1) is not obvious; its proof appears in [9]. The top-down construction of the 2-HST in [9] is hard to implement in a distributed network.

### 4.1 Distributed algorithm to compute 2-HST

In this section we propose a bottom-up construction to compute a 2-HST in a distributed way, with total communication cost $O(n \log n)$. The HST construction is performed as preprocessing at the network initialization stage. Specifically, we will show how to compute signature vectors shown in (1) and the necessary information for each node to find path to its parent in a distributed way.

Our algorithm proceeds in a bottom-up fashion and computes the $i$-th element of the signature vector for every node in round $i$, for $i$ starting at 0. If $S(u)_i = v$, we say that $u$ *nominates* $v$ at round $i$. Intuitively, as $i$ increases, only the nodes with small indices can possibly be nominated and appear in the signature vector. Observe that if at level $i$ a node $u$ is not nominated by any other node, i.e., every node $v$ already has some other node with index smaller than $u$ within radius $2^i\beta$, the node $u$ cannot possibly be nominated at level $i+1$. Thus we keep track of the subset of nodes that have been nominated and 'survive' round $i$ and only these nodes will flood the network with maximum hop count (TTL) value of $2^{i+1}\beta$ in round $i+1$. We use $P_i$ to denote the nodes that are nominated from the round $i$ and thus are candidates to be nominated in round $i+1$.

At the beginning, every node is a candidate, that is $P_0 = P$. A fixed pre-determined node ('leader') chooses a random $\beta$ uniformly from $[\frac{1}{2}, 1)$ and distributes it to all nodes using a single global flood. In round $i+1$, the nodes of $P_i$ flood the network up to distance $2^{i+1}\beta$. The flooding packets are cached at the nodes receiving these packets until the end of the current round. Now every node $u$ receives some flooding messages from the candidate nodes $P_i \cap B(u, 2^{i+1}\beta)$. During round $i+1$, node $u$ maintains the identity of the node $v_{\min}$ with lowest rank that it has received. At the end of round $i+1$, $u$ nominates $v_{\min}$ as its $(i+1)$-th level element in its signature vector for round $i+1$ (also its level-$(i+1)$ ancestor), i.e., $S(u)_{i+1} = v_{\min}$.

Each node records the information where it got the flooding message from. With this information, each node $u$ can trace back to get the path to the ancestor $v_{\min}$ and report the nomination. By this process, each node knows its parent and its children in that level of the tree. At the end of each round, all nodes clear all flooding messages that are not traced back by other nodes. $P_{i+1}$ consists of the nodes in $P_i$ that are nominated by some nodes during round $i+1$. Continue the above process until there are no more candidates.

We remark that the 2-HST is computed and stored in a completely distributed manner. This is sufficient for the applications in the distributed matching problem.

## 4.2 Communication cost

In this section we prove that the total communication cost, measured by the total number of message transmissions, for the computation of the 2-HST described earlier is bounded by $O(n \log n)$, provided that the doubling dimension of our minimum hop metric is bounded by a constant. The constant doubling dimension metric has been used in prior work as an appropriate model of a sensor network metric, when the sensor nodes are densely and uniformly deployed in a geometric region [10].

First we formalize the intuition stated above, that the sets of candidates $P_i$ become sparser as level $i$ increases. As the construction algorithm is randomized, the following bounds are taken in expectation on random permutations and parameters $\beta$. Actually, communication cost analysis in this section holds for any fixed $\beta$ (the fact that $\beta$ is random is only important for the distortion bound).

**Lemma 4.2.** *If the doubling dimension is at most $\gamma$, any ball of radius $2^i$ contains at most $2^{6\gamma}$ elements of $P_i$ in expectation, for all $i$.*

PROOF. For convenience, let $2^{6\gamma} =: c$. The proof is by induction on $i$. The claim is true for the first round ($i = 0$) as every node sends a message to its 1-hop neighbors, and each node only responds by one message to the neighbor with minimum index.

Consider the ball $B(v, 2^i)$ for arbitrary $v$. Nodes of $P_i$ in this ball may only be nominated (in the $i$-th round) by nodes in $B(v, 2^{i+1})$. The latter can be covered by a family $\mathcal{A}$ of $2^{2\gamma}$ balls of radius $2^{i-1}$, so it suffices to prove that each ball of $\mathcal{A}$ nominates at most $2^{-2\gamma}c$ nodes in $P_i$.

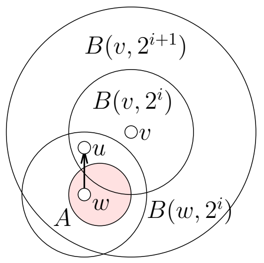

**Figure 1.** Any ball of radius $2^i$ contains at most $2^{6\gamma}$ elements of $P_i$ in expectation, for all $i$. The arrow from $w$ to $u$ means node $w$ nominates $u$ in round $i$.

Let $A$ be any of the covering balls, $A \in \mathcal{A}$. Denote by $|A|$ the number of nodes of $P$ inside $A$. If a node in $P_i \cap B(v, 2^i)$ is nominated from $A$, it must be chosen from $P_{i-1} \cap B(v, 2^i)$. See Figure 1 for illustration. Since $B(v, 2^i)$ can be covered by at most $2^\gamma$ balls of radius $2^{i-1}$, and by inductive hypothesis, there are at most $2^\gamma c$ choices for this nomination. That is, $|P_{i-1} \cap B(v, 2^i)| \le 2^\gamma c$. We claim that for a fixed choice $u$, the probability that $u$ is nominated by a node in $A$ is at most $\frac{1}{|A|}$. To see this, note that any $w \in A$ contains entire $A$ inside its $2^i$-ball, that is $A \subseteq B(w, 2^i)$. So if $u$ is nominated by any node in $A$, then in particular, $\pi(u)$ is smaller than the ranks of *all* nodes in $A$, which is true for at most $\frac{1}{|A|}$-fraction of permutations. Hence in expectation, the number of nominations by $A$ is $\frac{2^\gamma c}{|A|}$. On the other hand, it is also deterministically bounded by $|A|$, as each node in $A$ can nominate only once. Hence the number of nominations by $A$ is bounded by $\min\{|A|, \frac{2^\gamma c}{|A|}\} \le \sqrt{2^\gamma c}$. This is at most $2^{-2\gamma}c$ by our choice of $c$. □

Now we can prove the main result.

**Theorem 4.3.** *For a sensor network with $n$ nodes, the total communication cost for constructing the 2-HST is $O(n \log n)$ in expectation, and each node uses expected storage space for $O(\log n)$ node IDs.*

PROOF. The initial flood (to dsitribute the value of $\beta$) takes $O(n)$ messages, because the maximum degree of the network is bounded. Now it suffices to prove that each node at each round receives $O(1)$ messages. The communication cost will then be only $O(n)$ in each round as there are $n$ nodes in total. Since there are $O(\log n)$ rounds, the total number of messages is $O(n \log n)$.

The number of messages transmitted (propagated) by node $u$ in round $i$ is at most $|P_{i-1} \cap B(u, 2^i\beta)| \leq |P_{i-1} \cap B(u, 2^i)|$. Since $B(u, 2^i)$ can be covered by at most $2^\gamma$ balls of radius $2^{i-1}$, each of which contains at most $2^{6\gamma}$ elements of $P_{i-1}$ (by Lemma 4.2), we conclude that $u$ transmits at most $2^{7\gamma} = O(1)$ messages in each round.

Total storage requirement consists of all next-hop pointers that encode the paths that realize edges of the HST. Notice that each such pointer can be charged to a unique flooding packet reception event. Hence the storage requirement for a given node $u$ is at most the number of flooding messages that $u$ received throughout the computation. Again, Lemma 4.2 implies that this number is $O(1)$ per round, or $O(\log n)$ in total. $\square$

## 4.3 $\alpha$-HST

The algorithm described above computes a 2-HST for a distributed network. It is not hard to convert a 2-HST to an $\alpha$-HST for any value $\alpha \geq 2$. We remark that converting a 2-HST to an $\alpha$-HST can also be done in a distributed setting. Basically at the end of our distributed algorithm, each node identifies its ancestors in the 2-HST $T$. Now we condense the tree $T$ to an $\alpha$-HST $T'$ by removing some intermediate levels of $T$ and re-connecting the nodes directly to their lowest ancestors that are not removed. In particular, the leaf nodes remain the same. For $i$ increasing from 0, suppose we have constructed the tree $T'$ up to level $i$. The nodes on level $i$ of $T'$ correspond to the nodes on level $j$ on tree $T$. Now we proceed to build the tree $T'$ in level $i+1$. To do that, we take the lowest level $j'$ in the tree $T$ such that the distance $\sum_{h=j}^{j'-1} 2^h = 2^{j'} - 2^j$ is no greater than $\alpha^i$ and remove all the internal nodes on level $j+1, j+2, \cdots, j'-1$ of tree $T$. The nodes on level $j'$ are nodes on level $i+1$ in tree $T'$. The nodes in level $i$ connect to their corresponding ancestors on level $i+1$ by a single edge with weight $\alpha^i$. See Figure 2 for an example. Notice that the tree metric $T'$ again dominates the original metric as we are only relaxing the edge weights. In particular, the edge connecting a node on level $i$ in $T'$ to its parent has weight $\alpha^i$, with $2^{j'} - 2^j \leq \alpha^i < 2^{j'+1} - 2^j$ by construction, where $j$ is level $i$'s corresponding level in $T$ and $j'$ is level $(i+1)$'s corresponding level in $T$.

The relaxation of the edge weights of $T'$ will add at most a constant multiplicative factor on the distortion as shown below. For any two leaf nodes $u, v$ on $T$, suppose their lowest common ancestor is located at level $i$ on $T'$ and level $j''$ on $T$. Clearly $j \leq j'' < j'$. Now we would like to bound $d_{T'}(u,v)$ by $d_T(u,v)$. First $d_T(u,v) = 2 \cdot \sum_{h=0}^{j''-1} 2^h = 2(2^{j''} - 1)$, $d_{T'}(u,v) = 2 \cdot \sum_{h=0}^{i-1} \alpha^h = 2(\alpha^i - 1)/(\alpha - 1)$. Since $\alpha^{i-1} < 2^{j+1}$. We have

$$d_{T'}(u,v) \leq \frac{2\alpha}{\alpha-1} d_T(u,v) + \frac{4\alpha - 2}{\alpha - 1} \leq 7 d_T(u,v).$$

That is, the distortion for $d_{T'}$ is also bounded by $O(\log n)$.

This conversion can be executed by each node in the network examining its signature vectors and selecting subset of the elements in exactly the same way. In the end we have an $\alpha$-HST that is again implicitly stored on the nodes in the network, and $O(\log n)$-probabilistically approximates the underlying network metric.

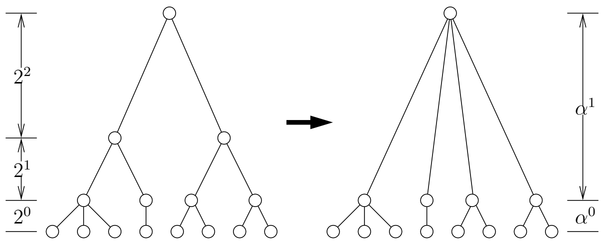

**Figure 2.** Convert a 2-HST to an $\alpha$-HST for any $\alpha \geq 2$.

At the end we summarize the properties of the $\alpha$-HST that will be useful later.

**Lemma 4.4.** *Suppose for level $i$ on the $\alpha$-HST $T'$ corresponds to level $\ell(i)$ on the 2-HST $T$. Now we have:*

1. $2^{\ell(i+1)} - 2^{\ell(i)} \leq \alpha^i < 2^{\ell(i+1)+1} - 2^{\ell(i)}$.
2. *Each leaf node is within distance $(\alpha^i - 1)\beta/(\alpha - 1)$ from its $i$-level ancestor on $T'$.*

PROOF. The first claim is due to the construction of the $\alpha$-HST. For the second claim, we know each leaf node is within distance $2^j\beta$ of its $j$-level ancestor on the 2-HST. Now the distance from a leaf node to its $i$-level ancestor on $T'$ is at most $2^{\ell(i)}\beta \leq [\alpha^{i-1} + 2^{\ell(i-1)}]\beta \leq (\alpha^i - 1)\beta/(\alpha - 1)$. $\square$

## 4.4 HST on $k$ resources

In Section 5 we will need to construct an HST only on the sub-metric of the original shortest-path metric which is induced by a given subset $K$ of $k$ nodes, let's call them *special* nodes for now. This can be done with the same communication and storage cost as before, by simply picking the permutation uniformly at random *from the set in which the special nodes are $k$ lowest-ranked nodes*, and in the end taking the part of the tree containing special nodes (notice that this is still a tree, and that it has the same root[2]).

[2]The parent of a special node is always special, due to the choice of ranks. Notice that the parent of a node could be itself if it has the lowest rank among all the nodes inside its proper neighborhood. Hence a breadth-first search from the root, stopped upon encountering a non-special node, is guaranteed to discovers special nodes.

Why does this work? The main observation is that the execution of the algorithm restricted to $K$ does not depend on non-special nodes at all. In fact, this is simulates the execution of the original algorithm of [9] on the metric $(K, d_K)$, where $d_K$ is the restriction of the shortest path metric on $K$. Recall that the same construction works for *any* metric, as long as the $2^i\beta$-neighborhoods are determined with respect to the appropriate distance function. Since this is the case here, and the metric has $k$ points, correctness follows.

## 4.5 Sparse aggregation with HST

We also remark that the 2-HST computed in Section 4 can be used to get an aggregation tree when spontaneously emerging resources spread out in the network are detected by their local sensors and no resource is aware of how many and where the other resources are. That is, it solves the same problem as in the sparse aggregation framework [11].

Suppose the a set $K$ of $k$ resources are detected simultaneously (i.e., at network setup), each node with new detection sends a message along the path in the tree towards the root. With the same assumption as in [11] that the messages travel with speed lower-bounded by $V$, these messages can be aggregated and pruned when they travel towards the root. In particular, an internal node $u$ at level $i$ will wait for a period of time $2^i \cdot V$ after the network setup — so that all messages from the resources in its subtree will be able to arrive at $u$ for sure. After that $u$ aggregates the messages it received and sends a message to its parent on the tree.

Eventually the process will result in a sparse aggregation tree $T(K)$ with root as the lowest common ancestor $LCS(K)$ of all the resources in $K$. The edges of $T(K)$ include the paths from the $k$ resources to $LCS(K)$. From $LCS(K)$ one message travels to the root (as it does not know whether there are resources in other part of the tree and has to travel to the root to find that out). The total communication cost of the sparse aggregation is $O(|T(K)| + \ell)$, where $|T(K)|$ is the size of the sparse aggregation tree, $\ell$ is the distance from $LCS(K)$ to the root of the HST and is upper bounded by the network diameter. See Figure 3 for an example. The blue nodes are the resources. The thick edges are the edges traveled by aggregation messages.

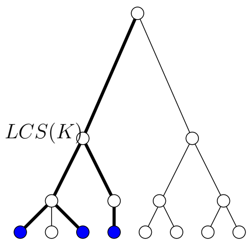

**Figure 3.** Sparse aggregation of $k$ resources/events by an HST.

We claim that the tree $T(K)$ has total size $O(\log k)$ times the smallest aggregation tree possible (i.e., minimum Steiner tree) in the original network (if the resources know each other in advance). In particular, $T(K)$ is the minimum Steiner tree of the resources in $K$ on the HST metric. Suppose the minimum Steiner tree in the original network is $T'(K)$, we replace each edge $xy$ on $T'(K)$ by the path between nodes $xy$ on the HST, with a blowup factor of $O(\log k)$, using Section 4.4. Now the resulting network must have a size no smaller than the size of $T(K)$ — as $T(K)$ is the minimum Steiner tree on the HST. This shows that $|T(K)| \leq O(\log k \cdot |MST|)$, where $MST$ is the minimum Steiner tree in the original network.

We remark that the paper [11] requires a double rulings scheme supported in the underlying network, for the probe messages to meet and discover each other. Our solution does not require that and can work on any network with bounded doubling dimension, given that the 2-HST has been computed in the preprocessing phase.

## 5. DISTRIBUTED MATCHING ALGORITHMS

In this section we describe several algorithms for distributed bipartite matching problem, in online and offline settings. The main goal of the section is to show that, if preceded by a preprocessing phase in which an HST is computed as in Section 4, the matching algorithms have *output-sensitive communication cost*, in addition to providing good approximation/competitive ratio (in offline and online setting, respectively).

In all models we consider there are $k$ resources whose number and locations do not change over time. The location of a given resource is initially unknown to any other node except the nearest sensor nodes that detect it. Also, we assume that there are exactly $k$ requests, which can either be present in the beginning, or arrive one by one. In any case, a perfect matching always exists. We remark that most of our results continue to hold if this is not the case.

We assume that packets traverse one hop per time unit, i.e., that appropriate algorithms are employed to handle low-level issues such as medium access, packet buffering and transmission scheduling. We also assume for simplicity that all nodes have enough local storage, and focus on minimizing communication cost. This is justified by the fact that with today's technology energy is a much more valuable resource than storage space.

## 5.1 Offline setting

First we consider the offline case, in which both resources and requests are present simultaneously (but the resources and the requests do not know each other). Bansal *et al.* [3] proved that the natural greedy algorithm — go through the set of requests in arbitrary order and match each request with the closest unmatched resource *in the $\alpha$-HST tree metric*, breaking ties arbitrarily — gives as $O(\alpha \log k)$-approximation (in expectation) to the optimal solution *in the original metric*, for any $\alpha > 1$. In the following we use a 2-HST and describe a distributed implementation of this algorithm that uses a factor of $O(\log k)$ more packets than the total length of the returned matching (the sum of hop-distances of matched pairs).

We may assume without loss of generality that no node has both a resource and a request; such pairs can be immediately paired and removed from consideration. We allow multiple resources to occupy the same node, the same with requests.

The distributed algorithm proceeds in a similar way as the sparse aggregation algorithm described in subsection 4.5. Resources and events send their information up the tree. Internal nodes match the resources and events in its subtree whenever possible. Information on unmatched resources or events is propagated further up the tree until every event is matched. Specifically, leaves that have resources or requests send this information to their parents (at level 1). Note that empty leaves do nothing. Parents then locally compute an arbitrary matching between requests and resources they received and notify matched leaves. We recursively solve the problem with unmatched requests and resources, where we think of them as residing at level-1 nodes. The resulting matching is represented by a number of paths in the HST, and corresponding paths that are constructed (by establishing 'next hop' pointers) in the underlying graph. Once this is done, level-1 nodes finish the algorithm by propagating each path to the leaf that initially holds the corresponding matched resource/request.

Notice that level-1 nodes have to wait at most one time unit for the packets coming from their children. This follows from out assumption on packet propagation speed, and 2-HST structure, i.e., the fact any leaf is within $2^1\beta < 2$ hops, hence at most one hop, away from its parent. If a packet from some child is not received within this time, it means that the child has no requests/resources to report. Similar time bounds hold in recursive calls.

Now we turn to bounding the communication cost. Clearly, no communication is ever 'wasted', i.e., if a node sends a packet to its parent reporting resources or requests (recall, never both), eventually that HST edge (and the corresponding path in the graph) will be used in the paths that match those resources/requests.

**Theorem 5.1.** *The expected communication cost of computing a matching of length $l$ with our algorithm is $O(\log k) \cdot l$.*

PROOF. The claim easily follows by observing that each edge computed by the algorithm may be a factor of $O(\log k)$ shorter in the original metric than in the HST metric, and by summing over all edges in the matching. □

## 5.2 Online setting

Now we turn our attention to the online setting in which requests come in one by one, and each request must be *irrevocably* matched to some resource before the next request arrives. This models fairly common applications of wireless networks in which decisions are time-critical, cannot be changed once they are made, and there is no *a priori* upper bound on the time between consecutive request arrivals.

We propose a distributed algorithm which never pays more than order of $l^\gamma$ in communication cost for computing a matching of cost (length) $l$, where $\gamma$ is the doubling dimension of the metric. Throughout this section we assume that the next request does not arrive until the current source is matched and all associated messages have been delivered. This assumption captures the application scenarios when event inter-arrival times are much greater compared with message propagation time.

One solution is based on the following randomized greedy algorithm proposed in a centralized setting by [15]. When a new request arrives, it is matched to the closest unmatched resource on an $\alpha$-HST, breaking ties randomly, i.e., by choosing each of the tied resources with equal probability. They proved that this algorithm is $O(\alpha \log^2 k)$-competitive (in expectation) on the original metric, provided that $\alpha \geq 1 + 2\ln k$.

For a node $u$, let $T(u,i)$ be the unique level-$i$ subtree that contains $u$. Also, for $i \geq 1$, define $N(u,i) = T(u,i) - T(u,i-1)$. Current request $r$ can be matched to its randomly chosen closest unmatched resource by having the leaf that holds $r$ gradually explore its neighborhood in the HST. Specifically, the exploration proceeds by levels: the first level is $N(r,0) = \{r\}$, then $N(r,1)$, then $N(r,2)$ etc. It is essentially a post-order traversal of the HST, which starts from $r$ and stops on encountering an unmatched resource. Notice that, with such organization, each $N(r,i)$ can be explored with a communication cost of at most twice the total length of all paths (in the underlying graph) that realize all edges of the subtree. This also includes the task of picking a uniformly random unmatched resource, if the subtree happens to contain any. For example, each resource, once found by exploration broadcast, can choose a random number uniformly from $[0,1]$, and then the minimum of all choices can be computed using sparse aggregation (as in Section 4.5). The 'trail' left by the minimum number's packet during aggregation leads to a uniformly random resource.

It is easy to see that the algorithm is correct (under the above assumption on request arrivals). It is also obvious that if $r$ is matched to $s$ after exploring levels up to $i$, i.e., entire $T(r,i)$, the communication cost involved is at most twice the total length of $T(r,i)$.

**Lemma 5.2.** *Consider an $\alpha$-HST constructed as in Section 4 for an underlying metric with doubling dimension $\gamma$. Let $T$ be a level-$i$ subtree and let $h$ be its height (root-to-leaf distance) in the HST metric. Total length of $T$ (all its edges) in the HST metric is at most $C(\alpha,\gamma)h^\gamma$, where $C(\alpha,\gamma) = \frac{\alpha^\gamma 2^{5\gamma}}{\alpha^\gamma - \alpha}$.*

PROOF. By the $\alpha$-HST construction in subsection 4.3 and Lemma 4.3, the root of the subtree $T$ is at level $\ell(i)$ at the corresponding 2-HST. Thus all leaves can be covered by a single ball of radius $2^{\ell(i)-1}\beta < 2^{\ell(i)-1}$ centered at the root of the subtree $T$. Now we consider a level $j$ in the subtree $T$ with $0 \le j \le i-1$ and we would like to count the number of nodes at level $j$. By Lemma 4.2, any ball of radius $2^{\ell(j)}$ has at most $2^{6\gamma}$ level-$j$ nodes. Hence the leaves can be covered by at most $2^{\gamma(\ell(i)-1-\ell(j))}$ balls of radius $2^{\ell(j)}$. Thus the number of level-$j$ nodes is at most $2^{\gamma(\ell(i)-\ell(j)+5)}$. Each level-$j$ node has a single edge of length $\alpha^j$ connecting it to its parent. Furthermore, the total length of the tree is exactly the sum of all these edges. So, summing over $j$, we have

$$\begin{aligned} \sum_{j=0}^{i-1} 2^{\gamma(\ell(i)-\ell(j)+5)} \cdot \alpha^j &= 2^{\gamma(\ell(i)+5)} \sum_{j=0}^{i-1} \alpha^{-(\gamma-1)\ell(j)} \\ &\le \frac{2^{\gamma(\ell(i)+5)}\alpha^\gamma}{\alpha^\gamma - \alpha} \\ &\le \frac{\alpha^\gamma 2^{5\gamma}}{\alpha^\gamma - \alpha} \cdot \left(\frac{\alpha^i - 1}{\alpha - 1}\right)^\gamma . \end{aligned}$$

The height of the subtree is $1+\alpha+\alpha^2+\cdots+\alpha^{i-1} = \frac{\alpha^i-1}{\alpha-1}$. Thus the total length of $T$ is bounded by $O(C(\alpha,\gamma)h^\gamma)$. $\square$

This directly implies a bound of $O(C(\alpha,\gamma)l^\gamma)$ on the communication cost of computing a single matching edge whose length is $l$ in HST metric.

We can now prove the main result about our online algorithm.

**Theorem 5.3.** *The expected communication cost for computing a matching of total length $l$ is $O(l^\gamma)$.*

PROOF. In expectation, a matched pair may be a factor of $O(\log k)$ closer in the original metric than in HST, yielding a ratio of $O(C(\alpha,\gamma)\log^\gamma k)$ for each edge. The same ratio holds for the total length, since $\sum_i x_i^\gamma \le (\sum_i x_i)^\gamma$, for $\gamma \ge 1$. The claim follows by substituting $\alpha = \Theta(\log k)$. $\square$

## 6. SIMULATIONS AND EXPERIMENTS

### 6.1 HST Construction

We implemented the 2-HST-construction algorithm (Section 4) in MATLAB and tested it on 100 randomly generated networks. The networks were generated by perturbing $n$ nodes of the $\sqrt{n} \times \sqrt{n}$ grid in the $[0,1]^2$ unit square, by 2D Gaussian noise of standard deviation $\frac{0.3}{\sqrt{n}}$, and connecting two resulting nodes if they are at most $\frac{3}{\sqrt{n}}$ apart (in Euclidean distance). We also made sure that all generated networks were connected.

Figure 4 illustrates a typical 2-HST layout on our test network, and Figure 5 shows a typical distribution of communication load and storage requirements. Communication is evenly distributed, as predicted by Lemma 4.2. Storage requirement is higher for nodes higher up in the hierarchy, but it grows slowly with the network size (Theorem 4.3). This result means that the hidden constant in Theorem 4.3 is in practice much smaller than predicted by our theoretical analysis.

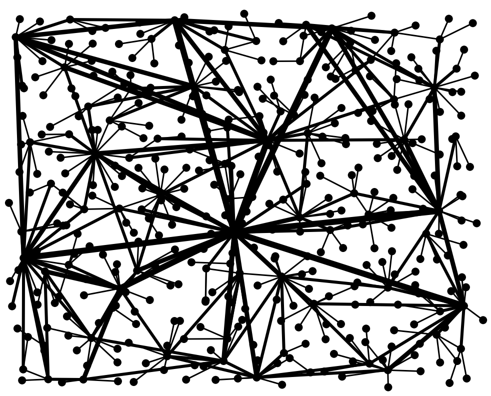
**Figure 4.** A typical HST on $n = 400$ nodes. Thicker lines represent higher-level edges.

Figure 6 shows how the communication cost and storage requirement change, on average, as a function of network size. The value for each size is an average of 100 independent trials. We conclude that, as predicted by Theorem 4.3, both per-node storage and per-node communication cost grow logarithmically with number of nodes.

### 6.2 Approximation and competitive ratios

We also implemented the offline algorithm described in Section 5.1. Since we know that it is communication-optimal in the 2-HST (Theorem 5.1), we only compared optimal costs of the matching in the 2-HST and in the underlying graph. This is, of course, directly related to the distortion of the HST embedding.

We generated random perturbed grid networks from the same distribution as in the previous experiment. Network sizes range from 25 to 400, and the number of resources was kept fixed at $k = 10$. We sampled 100 networks for each network size. For each network, we chose $k$ resources and $k$ requests uniformly at random, and computed a 2-HST *with requests and resources as special nodes* (Section 4.4). Then we computed the communication cost of the matching computed by our offline algorithm, the length (in hops) of the same matching in the $\alpha$-HST metric, and the length of the *optimal* matching on the same set of resources/requests (computed by the Hungarian algorithm [16], as implemented in [1]).

The following two figures show how the ratio of optimal matching costs (in 2-HST and the original metric) changes with the network size $n$ when the number of resources $k$ is fixed (Figure 7) and vice versa (Figure 8).

Finally, we consider the online algorithm of Section 5.2. We tested perturbed grid networks between 25 and 400 nodes,

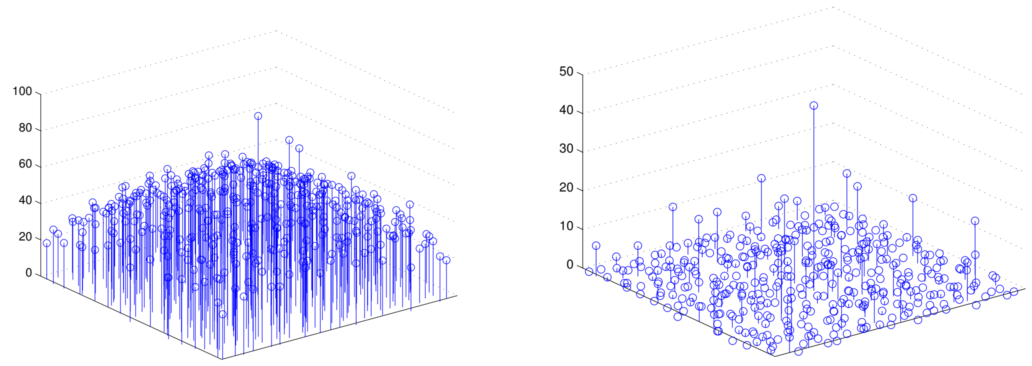

**Figure 5.** Communication cost (left) and storage requirement (right) in a perturbed grid network of $n = 400$ nodes.

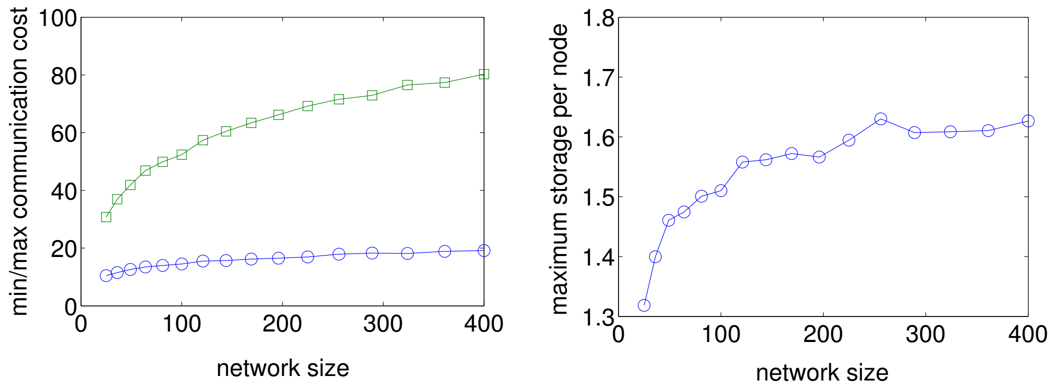

**Figure 6.** Per-node communication cost and storage requirement scale sublinearly with network size.

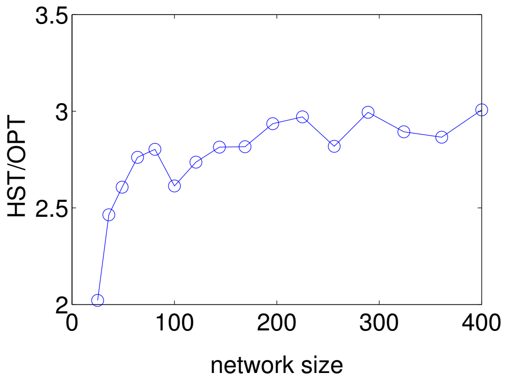

**Figure 7.** In offline setting, length of the optimal matching in the 2-HST and length of the optimal matching in the underlying metric are within a factor of 3 over a wide range of network sizes $n$, with fixed $k = 10$.

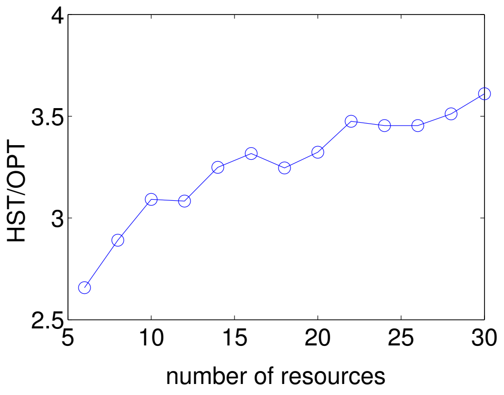

**Figure 8.** In offline setting, the ratio of optimal matching length becomes worse when the number of resources $k$ increases, and $n = 225$ is fixed. This is because the network embeds into 2-HST with distortion that increases with $k$.

with $50$ samples for each size according to the same distribution as above. For each sample we computed an $\alpha$-HST for $\alpha = 1 + 2\ln k$ and ran the algorithm, recording the resulting communication cost, cost of the matching returned by the algorithm (in the HST metric), and optimal cost in the original metric.

Figures 9 and 10 show the ratio of the communication cost and costs of optimal matchings, one in the HST and the other in the original metric. In Figure 9 the number of resources $k$ is fixed and network size changes. Figure 10 shows the opposite situation.

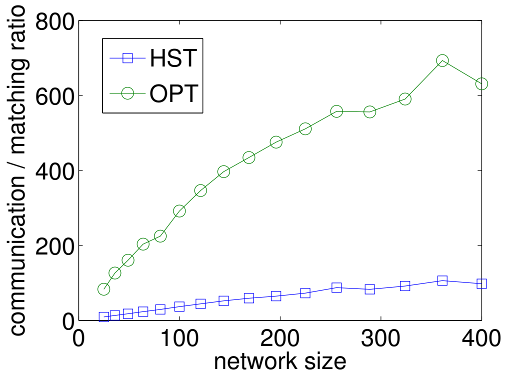

**Figure 9.** Communication overhead as a function of network size, with fixed $k = 15$.

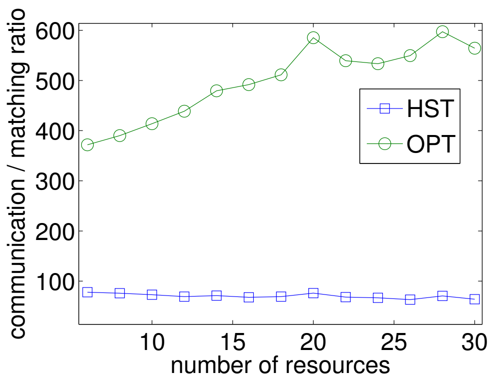

**Figure 10.** Communication overhead as a function of the number of resources, with fixed $n = 225$.

Figure 9 shows that, as predicted by Theorem 5.3, communication cost grows roughly quadratically with the cost of computed matching in the tree metric. This is expected, since we tested on two-dimensional geometric graphs.

## 7. CONCLUSION

Resource management in a distributed setting is a challenging problem due to the lack of global coordination and knowledge of available resources/emerging events. However, the problem can be simpler when the underlying network metric is simpler (such as a tree). This paper shows how to extract the hierarchically separated tree metric from a network in a distributed and communication efficient way, as well as how to implement the distributed resource management algorithms in the tree metric, for both offline and online scenarios. We would like to emphasize that the idea of working on a simpler metric can possibly be applied to other problems in a distributed setting to derive communication efficient algorithms with quality guarantees, such as $k$-server [6, 8], metric labeling [14], and tracking mobile sinks [2]. We will further explore this direction in the future.

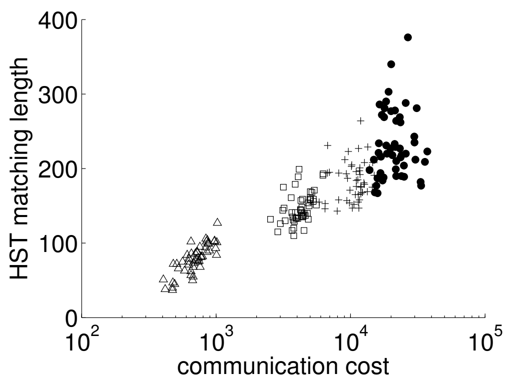

**Figure 11.** Communication cost of the online matching algorithm as a function of the cost of the computed matching in the $\alpha$-HST metric, with a fixed number of $k = 15$ resources and variable network size $n$. The slope suggests quadratic dependence (linear regression yields a slope of about 2.37).

**Acknowledgement:** Jie Gao and Dengpan Zhou acknowledge the support from NSF CAREER Award CNS-0643687. Leonidas Guibas and Nikola Milosavljević gratefully acknowledge the support of Army AHPCRC grant W911NF-07-2-0027-1 and NSF grants CNS-0626151, CCF-0634803.

## 8. REFERENCES

[1] http://www.mathworks.com/matlabcentral/fileexchange/6543/.

[2] B. Awerbuch and D. Peleg. Concurrent online tracking of mobile users. In *SIGCOMM '91: Proceedings of the conference on Communications architecture & protocols*, pages 221–233, New York, NY, USA, 1991. ACM.

[3] N. Bansal, N. Buchbinder, A. Gupta, and J. S. Naor. An $O(\log^2 k)$-competitive algorithm for metric bipartite matching. In *Proceedings of the 15th Annual European Symposium (ESA'07)*, pages 522–533, 2007.

[4] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *FOCS '96: Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, page 184, Washington, DC, USA, 1996. IEEE Computer Society.

[5] Y. Bartal. On approximating arbitrary metrices by tree metrics. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 161–168, New York, NY, USA, 1998. ACM.

[6] Y. Bartal and A. Rosen. The distributed $k$-server problem – a competitive distributed translator for $k$-server problems. In *Proceeding of IEEE Symposium*

*on Foundations of Computer Science*, pages 344–353, 1992.

[7] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 2005.

[8] A. Coté, A. Meyerson, and L. Poplawski. Randomized $k$-server on hierarchical binary trees. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 227–234, New York, NY, USA, 2008. ACM.

[9] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 448–455, New York, NY, USA, 2003. ACM.

[10] S. Funke, L. J. Guibas, A. Nguyen, and Y. Wang. Distance-sensitive information brokerage in sensor networks. In *Proceedings of the second IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 234–251, 2006.

[11] J. Gao, L. J. Guibas, J. Hershberger, and N. Milosavljević. Sparse data aggregation in sensor networks. In *Proc. of the International Conference on Information Processing in Sensor Networks (IPSN'07)*, pages 430–439, April 2007.

[12] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995.

[13] A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 534–543, 2003.

[14] J. Kleinberg and Éva Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. *J. ACM*, 49(5):616–639, 2002.

[15] A. Meyerson, A. Nanavati, and L. Poplawski. Randomized online algorithms for minimum metric bipartite matching. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 954–959, New York, NY, USA, 2006. ACM.

[16] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, Englewood Cliffs, NJ, 1982.

[17] E. M. Reingold and R. E. Tarjan. On a greedy heuristic for complete matching. *SIAM J. Computing*, 10(4):676–681, November 1981.