Maintaining Approximate Minimum Steiner Tree and k-center for Mobile Agents in a Sensor Network

Dengpan Zhou*   Jie Gao*

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

Abstract—We study the problem of maintaining group communication between m mobile agents, tracked and helped by n static networked sensors. We develop algorithms to maintain a $O(\lg n)$-approximation to the minimum Steiner tree of the mobile agents such that the maintenance message cost is on average $O(\lg n)$ per each hop an agent moves. The key idea is to extract a 'hierarchical well-separated tree (HST)' on the sensor nodes such that the tree distance approximates the sensor network hop distance by a factor of $O(\lg n)$. We then prove that maintaining the subtree of the mobile agents on the HST uses logarithmic messages per hop movement. With the HST we can also maintain $O(\lg n)$ approximate k-center for the mobile agents with the same message cost. Both the minimum Steiner tree and the k-center problems are NP-hard and our algorithms are the first efficient algorithms for maintaining approximate solutions in a distributed setting.

I. INTRODUCTION

In this paper we examine the challenges arising from a system of embedded static wireless nodes and a set of mobile agents acting in the same physical space. The agents could be either mobile robots or human users collaborating on a task. They do not necessarily have out-of-band communication channels among them and may have to resort to multi-hop routing in the static wireless network for inter-communication. The embedded sensor nodes provide real-time monitoring of the environment, possibly in-network processing to aid situation understanding, and interact with the mobile agents as a supporting infrastructure to help with both coordination and communication of the agents. This model captures many real-world scenarios, ranging from robots exploring a space, mules collecting sensor data, rescue team helping with disaster relief, etc.

We focus on two specific problems in this framework. The first is to maintain group communication of the mobile agents, in particular, a tree spanning the agents for continuous information exchange, decision making and order dissemination. We denote by the sensor closest to an agent i a *proxy node* for i that represents this agent. To reduce communication delay, the communication tree is preferred to be the minimum Steiner tree of the proxy nodes in the sensor network, i.e., the tree with minimum total hops connecting the proxy nodes, possibly by using other non-proxy nodes as relay. As agent i moves, its proxy node may hop to a neighboring node. In that case we will need to update the tree. It is well known that the minimum Steiner tree is an NP-hard problem [13], therefore we maintain at best an approximation to it. The objective is to achieve small update cost with a good approximation ratio to the optimal

minimum Steiner tree.

The second problem we study is the mobile k-center problem, that asks for k sensors as *centers* such that the maximum distance from each agent to its closest center is minimized. These centers naturally lead to spatial clustering or grouping of the agents, and map to obvious locations for control centers where information from the agents are aggregated and disseminated. In another scenario, when the agents need to physically gather together, the 1-center solution is the meeting point that minimizes the maximum travel distance. Again the k-center problem with k as a parameter is NP-hard [13]. We ask for the efficient maintenance of an approximate solution.

Challenges. There are two fundamental challenges to allow such coordination and communication between mobile agents.

The first problem is *location management*, that is, the tracking and dissemination of the current location of the agents. In our setting we identify the location of an agent with its proxy sensor node. Obviously the proxy node is aware of the agent in its proximity. The difficulty is to inform other nodes/agents of the current location of an agent. This problem has been studied extensively in the literature [1], [3], [9], [14]. Although being theoretically intriguing and inspiring, these schemes are still relatively heavy for real-world systems.

The second challenge is efficient maintenance of approximate minimum Steiner tree or approximate k-center of mobile agents, as efficient algorithms even in the centralized setting are lacking. Nothing is known for maintaining α-approximate minimum Steiner tree when $\alpha < 2$. The minimum spanning tree (MST) is a 2-approximation of the minimum Steiner tree. Yet the maintenance of MST is not efficient even in the centralized kinetic data structure framework [12] (i.e., when the moving trajectories of all agents are given) [2], [6]. In additional the centralized algorithms are very involved with heavy algorithmic techniques and are not practical. In the sensor network setting, a scheme called RoamHBA has been proposed to maintain group communication [8]. RoamHBA assumes that the current location of all agents are available (thus requiring a location management scheme as described above) and uses a heuristic algorithm to generate a tree. There is no guarantee on either the approximation to the MST nor the update communication cost when the agents move.

Regarding the k-center problem, the situation is not much better. In the centralized setting, the only known result is a kinetic algorithm for maintaining a 8-approximate k-center [10]. There is no distributed algorithm for k-center problem.

Our approach. We first preprocess the sensor network to extract a hierarchical tree metric H that approximates the original graph metric by a logarithmic distortion. Now, the minimum Steiner tree on a tree metric is trivial — it is simply the connection of edges from all the agents/proxy nodes to their common ancestor on H. As H is a $O(\lg n)$ approximation of the graph metric G, the minimum Steiner tree computed on H is a $O(\lg n)$ approximation of the MST on the graph metric G. Similarly, for the k-center problem, working on a tree is much easier than working with a general graph metric. One can compute the optimal k-center solution on H, which is a $O(\lg n)$ approximate solution for the k-center in G. With the preprocessing no location management is needed for MST and k-center maintenance; the agents are not aware of the current location of other agents. Thus we save the communication cost necessary to update and query for the current agent location.

The tree we extract is a 2-hierarchically well-separated tree H (2-HST) and it is randomized. Approximating a metric with probabilistic hierarchical well-separated trees was first proposed by Bartal [4], [5], with the motivation that many problems are easier to solve on a tree than on a general graph. Later, Fakcharoenphol *et al.* [7] improved the distortion to $O(\lg n)$ for any n node metric and this is tight. This previous works, however, were for a centralized environment. In an earlier work of ours [11], we developed a distributed algorithm to extract a HST in a sensor network setting. In [11], we also applied the HST to the problem of resource management and distributed matching. The application of HST on the minimum Steiner tree and k-center problem, as well as their maintenance when the agents are mobile, are new results.

In this paper we show that with the tree metric constructed in the preprocessing phase, we can maintain the MST and the k-center of the mobile agents with expected communication cost of $O(\lg n)$ for each hop the agents move. We also demonstrate the experimental results for our scheme and compare with the only previous scheme RoamHBA [8].

II. NETWORK SETUP

Our following discussion is based on the assumption of a static sensor network consisting of n sensor nodes P and m mobile agents S. Agents are tracked by nearby sensor nodes called proxy nodes. We assume that every node is aware of the number of mobile agents, m.

A metric (P, d) has growth rate γ, if for any $v \in P$ there are $c_1 \cdot r^{\gamma-1} \leq f(r) \leq c_2 \cdot r^{\gamma-1}$ nodes with distance exactly r from v, for some constant c_1 and c_2, $c_1 \leq c_2$. The doubling dimension of a metric space (P, d) is the smallest value λ such that every ball with radius r in P can be covered by λ balls of radius $r/2$. The doubling dimension of P is then defined to be $\dim(P) = \lg \lambda$. We call a metric to have a constant doubling dimension if its doubling dimension is bounded by a constant. It is not hard to see that a metric with bounded growth rate has constant doubling dimension, by a simple packing argument.

All sensors are uniformly distributed in a planar domain with nearby nodes directly communicating to each other.

The communication network on the sensors is denoted as $G = (P, E)$. In our case, the network metric (P, d_G) is the minimum hop count metric in the sensor network. The vertices in the metric are the sensor nodes. The distance between two nodes is the minimum hop count value. In the following when we mention the "network metric" or the "original metric", we mean the minimum hop count metric (P, d_G). We assume that the minimum hop count metric of the communication graph has bounded growth rate of 2, and therefore constant doubling dimension.

III. HST REVIEW

In this section, we review the definition of α-hierarchically well separated tree (α-HST) and a distributed algorithm to compute the α-HST for a doubling dimension metric. The nodes of the HST are the nodes in the sensor network. The edges of the HST are virtual edges and the weights are redefined.

Definition 3.1 (α-HST). *A rooted weighted tree H is an α-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 α as we go down the tree.*

Fakcharoenphol *et al.* [7] gives a centralized algorithm to compute a 2-HST metric d_H for an n-point metric (P, d_G). When we say the "tree metric", we mean the distance on the HST and denote it as d_H. d_H provides a $O(\lg n)$-probabilistically approximation on d_G, that is, for any $u, v \in P$, $d_H(u, v) \geq d_G(u, v)$, and $E[d_H(u, v)] \leq O(\lg n) \cdot d_G(u, v)$, where the expectation is taken over random choices of the algorithm. The algorithm is as follows: We uniformly choose a random permutation $\pi : P \rightarrow \{1, 2, \ldots, n\}$ of the nodes from the set of all permutations. We also fix a value β chosen uniformly from $[\frac{1}{2}, 1)$. For convenience, $B(u, r)$ denotes a ball with radius r centered at u, and D represents the network diameter. For each node u, we compute a $O(\lg n)$-dimensional *signature vector* $S(u)$, where the i-th element in the vector is $S(u)_i = arg \min_{v \in B(u, 2^i \beta)} \pi(v)$ for $i = 0$ to $\ell = \lceil \lg D \rceil + 1$. In other words, each node keeps the node with the smallest rank among all nodes within distance $2^i \beta$. $S(u)_\ell$ is the node with rank 1 for all nodes u. 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 $S(u)_i$, and the weights of all edges between level i and level $i - 1$ is 2^i. Figure 1(a) shows an HST example.
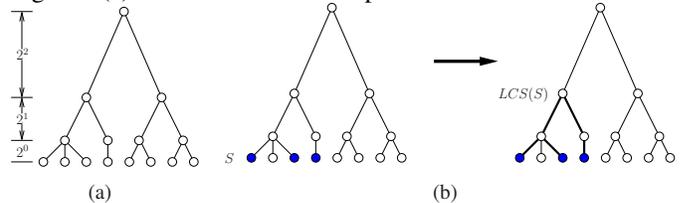
Fig. 1. (a) An example HST constructed from the algorithm. (b) An example to show how to get minimal spanning tree on HST for a subset agents.

Since this algorithm is centralized, in [11] we provide a distributed implementation of the HST construction. The

algorithm proceeds in a bottom-up fashion and compute the i-th element of the signature vector for every node in round i. The simple idea behind this is that, as i increases, only nodes with small ranks can be nominated and remained in the next round. The value of β, chosen uniformly at random from $[\frac{1}{2}, 1)$, is distributed to all nodes in the network by a single global flood. Initiatively, every node is a candidate, that is, $P_0 = P$. In round $i + 1$, the nodes remaining in P_i flood the network up to distances $2^{i+1}\beta$. The flooding packets are cached at each node receiving these packets until the end of each round. Then each node u in the network will choose the node v_{min} with the lowest rank among all the nodes it receives in this round, and nominate it as its $(i+1)$-th level element for its signature vector, i.e. $S(u)_{i+1} = v_{min}$.

At the end of the algorithm, each node u keeps a signature vector $S_u(k)$, where $S_u(i)$ is u's i-th level ancestor. This information is enough for constructing the HST and convenient for our application.

As the algorithm proceeds, fewer and fewer nodes remain active though the flooding range increases. Therefore the total message cost is still near linear. We have proved in [11] the following lemma.

Lemma 3.2. *For a sensor network with n nodes, the total communication cost for constructing the 2-HST is $O(n \lg n)$ in expection, and each node uses expected storage space for $O(\lg n)$ node IDs.*

For our applications, we list some of the properties of HST here.

Lemma 3.3 (α-HST properties). *Suppose the HST metric (H, d_H) corresponding to the original metric (P, d_G)*
1) *For any $u, v \in P$, $d_H(u,v) \geq d_G(u,v)$ and $E[d_H(u,v)] \leq O(\lg n) \cdot d_G(u,v)$.*
2) *For any $u \in P$, suppose its i-th level ancestor (from leaf to root) in H is u_i. We have $d_H(u_{i+1}, u_i) = \alpha \cdot d_H(u_{i-1}, u_i)$.*
3) *For $\forall i, j (1 \leq i, j \leq \ell, \ell$ is the HST height), the distance between all the nodes in level i and j are of the same value.*

IV. MAINTAINING APPROXIMATE MINIMUM STEINER TREE

In this section, we have a static wireless sensor network with node set P. We want to show how to maintain a minimal Steiner tree for a set of agents S ($m = |S| \leq n = |P|$), which reside on the sensor nodes and may move in the sensor network. First we compute a 2-HST H for the original metric network in a distributed way as stated in our previous section. Each node keeps its ancestors at each level. This process is implemented at the beginning of our application, and it will be used for all the following applications.

Compute approximate minimum Steiner tree. The minimum Steiner tree $H(S)$ connecting the agents is simply the collection of edges that connect them to their lowest common ancestor in H. In particular, each proxy node sends a counter with the number of agents residing in that node to its parent

on H. Each internal node generates a new counter by adding up the counters received from its children, and delivers it to its parent. When the counter value equals m, the corresponding internal node is the root of the minimal Steiner tree and the edges of $H(S)$ include all the edges on the path from the proxy nodes to w. This tree $H(S)$ is actually realized in the original graph G such that each edge uv in $H(S)$ is replaced with the shortest path connecting u, v. This gives us a Steiner tree $T(S)$ in the original metric[1]. Figure 1(b) shows the basic idea of the above algorithm. If the sensor nodes are not aware of the number of agents m in the network, we can solve this by sending notification to upper level until the HST root from the proxy nodes, then tracing back to the child of the lowest internal node u that gets only one notification from its children. And this node u is the root of the minimal Steiner tree.

Maintenance under agent motion. When an agent moves from one node p to a neighboring node q, we update the minimal Steiner tree on the HST. We only need to update the paths from p, q to their lowest common ancestor. Both p and q send a notification upward until their lowest common ancestor u in H. All the nodes along the path from p to u will decrease their counters by 1. Any node v with counter 0 will delete the edge to its parent in the tree. The nodes on the path from q to u will increase their counter by 1 and add the path to the Steiner tree $H(S)$ (as well as the Steiner tree $T(S)$ in the original metric).
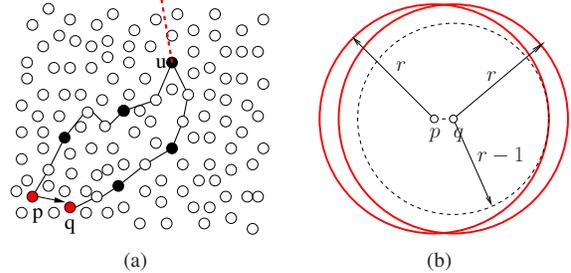
Fig. 2. 2(a): When a agent moving from p to q, we only repair the paths from p, q to their lowest common ancestor. 2(b): Suppose p, q have different common ancestors at level i. The lower rank one is not within distance r from both p, q.

Performance analysis. Since H approximates the metric in G with a maximum distortion $O(\lg n)$, it is easy to see the following theorem.

Theorem 4.1. *For a set of agents S, the minimum Steiner tree $T(S)$ is a $O(\lg n)$ approximation for the optimal minimum Steiner tree $MStT(S)$ in the original graph G.*

When an agent moves from p to a neighbor node q, and their lowest common ancestor w is at level i, the update cost is $O(2^i)$. In the worst case w can be the root of H and the update cost is the same as $\Omega(D)$, where D is the network diameter. However, the probability for such expensive update to happen is small. Specifically, at level $i-1$, p, q have different

---

[1]$T(S)$ is a logical tree on G and may contain duplicate edges and possible cycles. One can remove duplicate edges or cut open the cycles to obtain a real Steiner tree on S in the graph G. This operation will further reduce the weight of the tree.

ancestors. The lower rank one is not within distance $r = 2^{i-1}\beta$ from both p, q. See Figure 2(b). By the assumption of bounded growth rate, the chance for this to happen is $O(1/2^i)$. Thus the expected update cost is only $O(\lg n)$.

Theorem 4.2. *For a metric (P, d_G) with bounded growth rate and a HST H with metric (P, d_H), the expected update cost of the minimum Steiner tree on H for each hop an agent moves is bounded by $O(\lg n)$.*

V. MAINTAINING APPROXIMATE k-CENTER

Definition 5.1 (k-center). *Given a sensor network with metric (P, d_G), where d_G is taken on the shortest hop distance for any two node $p, q \in P$ in network G, for a set of agents (actually their proxy nodes) $S \subseteq P$ and an integer k, compute a node set $K = \{a_1, a_2, \ldots, a_k\} \subseteq P$ such that the maximum distance from any agent to its closest center, $\max_{p \in S} \min_{a \in K} d(p, a)$, is minimized.*

We show that with the same data structure for maintaining the minimum Steiner tree on the HST, we also maintain a $O(\lg n)$ approximate solution for k-center problem for any k. As stated before, each node will keep two types of information: a pointer to its parent on H and a counter that counts the total number of agents located in the subtree rooted at this node. For a set of agents S, the lowest common ancestor u of S will be a good candidate for the 1-center with a $O(\lg n)$ approximation. To find a good k-center solution, we simply take the lowest level on the HST such that the number of nodes with non-zero counter is no greater than k. These nodes are the k-centers. Again we can show that the approximation ratio is $O(\lg n)$ compared with the optimal k-center of G. The proof is omitted.

**Theorem 5.2.** *The k-center solution in the HST metric gives a $O(\lg n)$ approximation to the optimal k-center in G.*

The k-center structure can be used for data aggregation at these k centers. In particular, each agent sends its data upward along the HST tree. Internal nodes will take the data from the subtree, compute the aggregation, and deliver to its parent on the HST. This way, the information is naturally aggregated on the internal nodes. The above theorem says that, for each level of the HST the aggregation result is at a subset of aggregation nodes, such that the distance from all agents to these aggregation nodes is not far from the minimum possible.

VI. SIMULATION

In this section, we implemented the above algorithms in MATLAB and compared them with existing algorithms. The cost in our simulation is counted as the hop distance.

Approximate minimal Steiner tree construction. 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 $\frac{2}{\sqrt{n}}$ apart. Figure 3(b) shows an example of our algorithm.
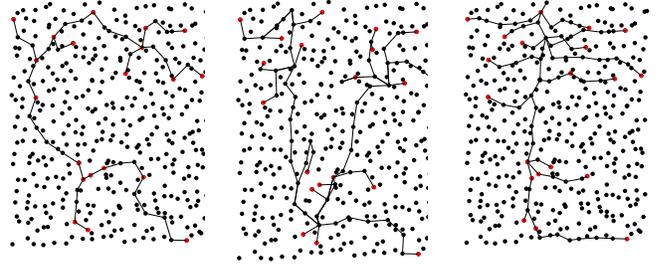
Fig. 3. 3(a): the minimum spanning tree. 3(b): The Steiner tree computed with our algorithm. 3(c): the Steiner tree computed with RoamHBA. The agents are in red. The network size is 400, and the agent size is 20.

Cost comparison with MST. We generated networks of size ranging from 400 to 2500, and the agent set size was kept at 100. We sampled 100 networks for each network size. For each network, we randomly generated an agent set of size 100 and compute an approximate minimal Steiner tree with our algorithm. We also computed the shortest path between each pair in the agent set. Then we had a complete graph $G(S)$. The vertices were the agent set, and the weight for each edge was the shortest hop distance in the original network. Now we computed the minimal spanning tree in this complete graph. After that, we got a Steiner tree in the original network, whose total weight is at most twice the cost of the minimum Steiner tree of S. In our following discussion, we use MST to denote the cost of this tree. Figure 3(a) is a minimal spanning tree for the same input as in Figure 3(b). From Figure 4(a), the cost
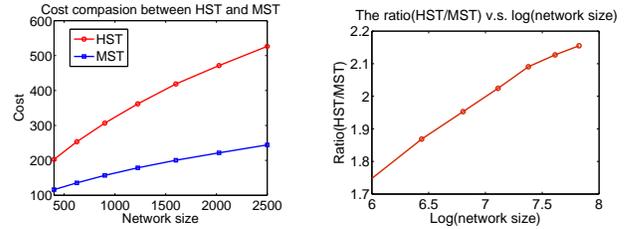
(a) Cost for different network size   (b) Cost ratio for log(network size)

Fig. 4. Cost comparison between HST and MST for different network size but the agent set size fixed to be 100.

of our algorithm is not far away from the MST case (within a factor of 2.2). The ratio increases slowly with the network size. According to Figure 4(b), the approximation ratio almost linearly depends on $\lg n$.

Now we fixed the network size to be 1000, and the agent set size ranged from 50 to 400. According to Figure 5(a) and
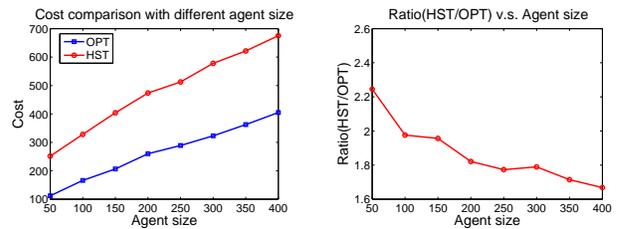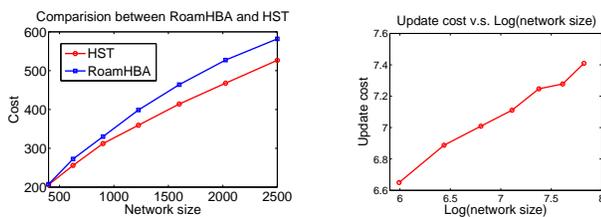

(a) Cost comparison.   (b) Cost ratio.

Fig. 5. The cost comparison between the HST and MST solution, with different agent set size but the network size fixed to be 1000.

Figure 5(b), the cost ratio between the HST and MST solution decreases when the number of agents increases.

Comparison with RoamHBA. In RoamHBA [8], the algorithm will choose a horizontal or vertical line across the network as the backbone. The backbone is constructed by choosing a median node with greedy forwarding in the desired direction. The rest of the agents connect to the backbone by greedy forwarding to the backbone. The greedy forwarding for node (x, y) is in the following way (assuming the horizontal direction): choose the neighbor that is nearest to point $(x + R, y)$ and $(x - R, y)$, R is the transmission range. Figure 3(c) shows an approximate minimal Steiner tree from RoamHBA from the same data as Figure 3(b) and Figure 3(a).

The network size ranged from 400 to 2500, and we choose a random agent set with size fixed at $k = 100$. We sampled 20 networks for each network size. Then we computed the approximate minimal Steiner tree from HST and RoamHBA respectively. Figure 6(a) shows that our algorithm is better

(a) Cost comparison between HST and RoamHBA solution.

(b) Update cost changes with log(network size).

Fig. 6. The cost is compared for HST-based solution and that by RoamHBA.

than RoamHBA with respect to the cost of the tree. But when the agent size is small, RoamHBA may have some advantage.

Update cost. The network size changed from 400 to 2500, and the agent size was kept fixed at $k = 100$. We sampled 15 networks for each network size. And we set a random moving direction for each agent. In each time step, the agent moved to $r/2$ (r is the communication range.) away along its moving direction. When an agent hit the network boundary, it bounded back along the reflected direction. In our simulation, we let all the agent move 1 step one by one. And repeated this 100 times. The we took the average of each time step and each node for the update cost for that network size.

From Figure 6(b), we see that the update cost almost changes linearly with logarithm of the network size. In fact, the update cost for each movement step is on average only $6 \sim 7$.

1-center. We generated networks with size from 500 to 2500, and chose $m = 100$ random agents for each network. We sampled 100 networks for each network size. We computed the approximate and optimal solution for 1-center, and compared them. Figure 7(a) gives the cost comparison. Figure 7(b) shows how the ratio changes with logarithm of the network size. We can see the nearly linear relationship between them.

VII. Conclusion

We show in this paper that the hierarchical well-separated tree, extracted from the underlying network, can be useful in maintaining $O(\lg n)$ approximate solution for the minimum Steiner tree problem or the k-center problem. In particular, the
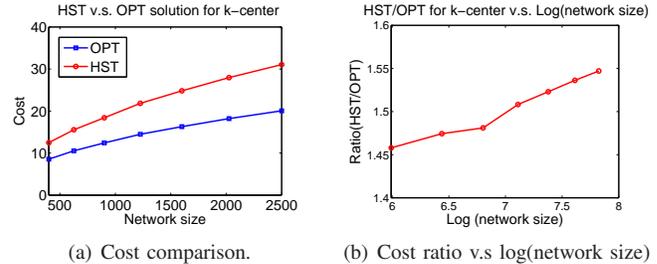
(a) Cost comparison.

(b) Cost ratio v.s log(network size)

Fig. 7. Cost comparison between HST and OPT for 1-center, with varied network size but agent size fixed to be 100.

agents do not need any information about the location of other nodes. The algorithm is also simple and distributed. This leads to applications such as maintaining group communication and aggregation nodes among a set of mobile agents. We would like to explore more applications of the hierarchical well-separated tree in the future.

References

[1] I. Abraham, D. Dolev, and D. Malkhi. LLS: a locality aware location service for mobile ad hoc networks. In *DIALM-POMC '04: Proceedings of the 2004 joint workshop on Foundations of mobile computing*, pages 75–84, 2004.

[2] P. K. Agarwal, D. Eppstein, L. J. Guibas, and M. R. Henzinger. Parametric and kinetic minimum spanning trees. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 596, Washington, DC, USA, 1998. IEEE Computer Society.

[3] B. Awerbuch and D. Peleg. Concurrent online tracking of mobile users. *SIGCOMM Comput. Commun. Rev.*, 21(4):221–233, 1991.

[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] J. Basch, L. J. Guibas, and L. Zhang. Proximity problems on moving points. In *SCG '97: Proceedings of the thirteenth annual symposium on Computational geometry*, pages 344–351, New York, NY, USA, 1997. ACM.

[7] 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.

[8] Q. Fang, J. Li, L. Guiba, and F. Zhao. Roamhba: maintaining group connectivity in sensor networks. In *IPSN '04: Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 151–160, New York, NY, USA, 2004. ACM.

[9] R. Flury and R. Wattenhofer. MLS: an efficient location service for mobile ad hoc networks. In *MobiHoc '06: Proceedings of the seventh ACM international symposium on Mobile ad hoc networking and computing*, pages 226–237, 2006.

[10] J. Gao, L. Guibas, and A. Nguyen. Deformable spanners and their applications. *Computational Geometry: Theory and Applications*, 35(1-2):2–19, 2006.

[11] J. Gao, L. J. Guibas, N. Milosavljevic, and D. Zhou. Distributed resource management and matching in sensor networks. In *Proc. of the 8th International Symposium on Information Processing in Sensor Networks (IPSN'09)*, pages 97–108, April 2009.

[12] L. Guibas. Kinetic data structures. In D. Mehta and S. Sahni, editors, *Handbook of Data Structures and Applications*, pages 23–1–23–18. Chapman and Hall, CRC, 2004.

[13] D. S. Hochbaum, editor. PWS Publishing Company, Boston, MA, 1997.

[14] J. Li, J. Jannotti, D. Decouto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proceedings of 6th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 120–130, 2000.