Low-Stretch Greedy Embedding Heuristics

Andrej Cvetkovski
Department of Computer Science
Boston University
Email: andrej@cs.bu.edu

Mark Crovella
Department of Computer Science
Boston University
Email: crovella@cs.bu.edu

***Abstract*—Greedy embedding is a graph embedding that makes the simple greedy packet forwarding scheme successful for every source-destination pair. It is desirable that graph embeddings also yield low hop stretch of the greedy over the shortest paths. In this paper we study how topological and geometric properties of embedded graphs influence the hop stretch. Based on the obtained insights, we construct embedding heuristics that yield minimal hop stretch greedy embeddings.**

I. Introduction

In the greedy routing paradigm (studied initially in [1], [2], [3]), the communication network is first embedded in a metric space by assigning to each node a coordinate denoting its location. From these coordinates, the straight-line geometric distances between any pair of nodes can be calculated. According to the simplest deterministic greedy routing rule, a node forwards a message to a directly connected neighboring node so that the new message location would reduce and locally minimize the distance of the packet to the destination.

Notable advantages of greedy routing are its small computational complexity, small memory requirement per node, and the use of local information only – each node finds a next hop based on the coordinates of its neighbors. As can be demonstrated by simulations, greedy routing based on node locations and distances in Euclidean space can often be successful, but is not guaranteed to be successful: it is easy to construct examples where a packet reaches a node that is closer to the destination than all of its direct neighbors so that the forwarding fails even though a path to the destination exists.

Routes found by greedy forwarding typically take more hops than the corresponding shortest paths. For each source-destination pair, we define the *hop stretch* as the ratio of hop lengths of the greedy path to the corresponding shortest path in the graph. The *average* and *maximum hop stretch* can be used as measures of the hop stretch for the entire graph.

Both the rate of success and the hop stretch of the forwarding depend on the graph topology but also on the node locations. Thus, for a given topology, we can speak of the success rate or the average hop stretch *of the embedding*. When the success rate is guaranteed to be 1 (100%) we call the embedding *greedy* [4].

If physical node coordinates are used for forwarding, the success rate and hop stretch can not be influenced readily. This motivates the study of network embeddings based on virtual coordinates [5], chosen so as to optimize success rate, hop stretch or some other quantity of interest (see also e.g. [6], [7] and the references therein.) Limiting the focus to metric spaces, [8] presented a constructive proof that every finite, connected, undirected graph has a *greedy embedding* in two-dimensional hyperbolic space.

In this work, we study the possibilities for constructing graph embeddings that are both greedy and have hop stretch as low as possible. Allowing virtual coordinates, the procedure of [8] can be used, and the resulting embedding will be greedy. Henceforth, we refer to the procedure outlined in [8] as *K-embedding*. Motivated by the observation that varying some of the parameters of the K-embedding can cause significant changes to the obtained average and maximum hop stretch, we study how the properties of the K-embedded graphs influence the hop stretch. We use the obtained insights to propose embedding heuristics that yield low hop stretch greedy embeddings. Our heuristics can be applied to any greedy embedding procedure based on the extraction of an arbitrary spanning tree, including those described in [8], [9], [10], [11]. To the best of our knowledge, ours is the first study to demonstrate these insights and to construct corresponding heuristics.

The rest of this paper is organized as follows. Section II provides more specific background on the problem at hand. We present the main ideas of our approach in Section III and further examine its qualities in Section IV. Technical details of the implementation are relegated to Section V. The current conclusions are in Section VI.

II. Preliminaries

The input to the K-embedding algorithm [8] is a finite, connected, undirected and unweighted graph representing a communication network. The K-embedding places the graph nodes in the hyperbolic plane, and uses the standard hyperbolic distance (see e.g. [12]) for the forwarding function. The generic algorithm of [8] finds a greedy embedding of the infinite d-regular tree for any integer $d \geq 3$. To embed an actual connected graph G, first *a spanning tree T of G is chosen* and d is determined as the maximum degree of any node in T. Subsequently, the nodes of T are *mapped to* the embedded nodes of the d-regular tree. The author in [8] proves that the result is a greedy embedding of T. It is easy to see that a greedy embedding of T is also a greedy embedding of the graph G.

To start our study, we note that the average node degree of the input graph, as a topological property, strongly influences the average hop stretch of any greedy embedding. In the extreme case when the graph is itself a tree ($G = T$), there is

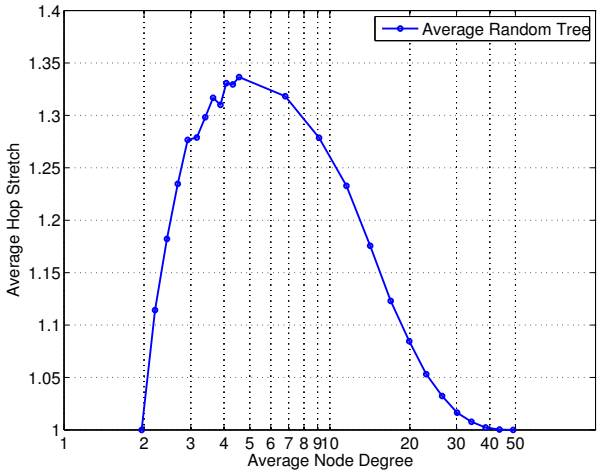

Figure 1. Dependence of the average hop stretch of a greedy embedding on the average node degree of the graph, averaged over 32 instance of a $G(n_0, p)$ graph. There is a range of critical node degrees, typically between 3 and 8 for which the hop stretch is maximal.

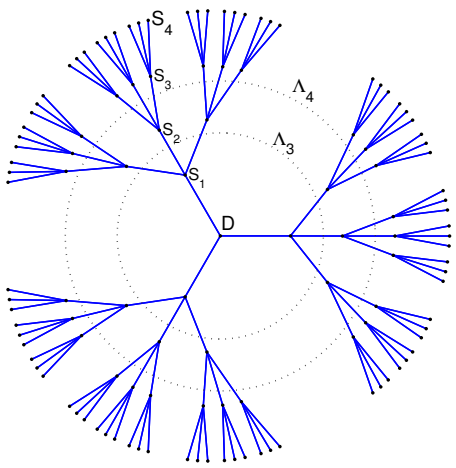

Figure 2. A simplified model: The number of next hop candidates exponentially decreases as the packet gets closer to the destination.

a unique path for each pair of vertices making the greedy and the corresponding shortest path coincide, and the hop stretch is 1. The other extreme case, when G is a complete graph, also has an ideal hop stretch of 1 since in this case all pairs are direct neighbors and both the greedy and the shortest paths are always 1 hop. Graphs with average node degrees between a bare tree and a complete graph typically have larger hop stretch.

To demonstrate the dependence of the hop stretch on the average node degree, we show an initial series of experiments using the largest connected component of $G(n_o, p)$ graphs with varying edge probability p, resulting in graphs with $n \approx 50$ nodes. For several values of the average node degree between those of a tree ($2(n-1)/n \lessapprox 2$) and a complete graph ($2n(n-1)/(2n) = n-1 \approx 50$) we perform a K-embedding based on a spanning tree sampled uniformly at random [13], and average the hop stretch over 32 graph instances. The results are shown in Fig. 1. The figure shows that there is a range of critical node degrees, roughly between 3 and 8 for which the hop stretch is maximal. In light of this fact, our subsequent experiments focus on node degrees from the critical range $[3...8]$.

While the hop stretch depends on both the choice of spanning tree and its mapping to the embedded nodes of the d-tree, we find that in practice the mapping step has relatively little impact. To illustrate this observation, we show results from a typical experiment in which for a randomly generated graph and a randomly chosen spanning tree, we generate 600 different random spanning tree mappings. For each mapping, we record the average and the maximum hop stretch of the embedding. As typical values, the average hop stretch is 1.28 ± 0.02 and the maximum hop stretch is 4.3 ± 0.26. Since in our experiments, the spanning tree mapping does not significantly influence the hop stretch of the embedding, in the remainder of our study, we focus only on the choice of spanning trees for low hop stretch embeddings.

III. Heuristics

A. Maximum Weight Spanning Tree Heuristics

We start by laying the foundation for our first heuristic, which we term the *maximum weight spanning tree* (MWST). Consider a connected graph G that is K-embedded [8] starting from some chosen spanning tree. The choice of a spanning tree T partitions all graph edges into *tree and non-tree (shortcut) edges*. A greedy path typically consists of some tree edges and some shortcut edges. Intuitively, shortcut edges may help make progress toward the destination more rapidly than the tree edges and would thus lower the hop-stretch.

When can a packet take a shortcut? Consider a packet currently at node S and destined for node D. There are between 1 and d tree edges incident to S, one of which leads to the relative parent of S (denoted $\pi(S)$) when D is taken as the root of the tree T. Forwarding to $\pi(S)$ certainly brings the packet closer to the destination D by the greedy property of the embedding of T. In addition, there may or may not be some non-tree edges incident to S. Of those non-tree edges, useful as a next hop will be only those that bring the packet closer to D than $\pi(S)$ is. The analysis of the exact shape of the locus of points containing next hop node candidates reachable via shortcuts from S is beyond the scope of this work, but to gain some insight, here we resort to a simplified calculation in which we consider the K-embedding of a graph whose spanning tree is the full regular tree of degree d.

As a concrete example, Fig. 2 schematically illustrates the first $L = 5$ levels of a regular tree with $d = 4$ spanning a random graph on n nodes along with several possible packet source locations (S_1 ... S_4), and a single destination D. Assuming that the graph has an average node degree $\bar{\delta} = 6$ leaves on average 2 non-tree edges incident on each node (not shown). The locus of next hop shortcut candidates when the packet is at node S_ℓ at level ℓ is labeled Λ_ℓ in the figure and contains approximately $\#\Lambda_\ell = n/d^{(L-\ell+1)}$ nodes. The probability of a non-tree edge between S_ℓ and any other node is assumed to

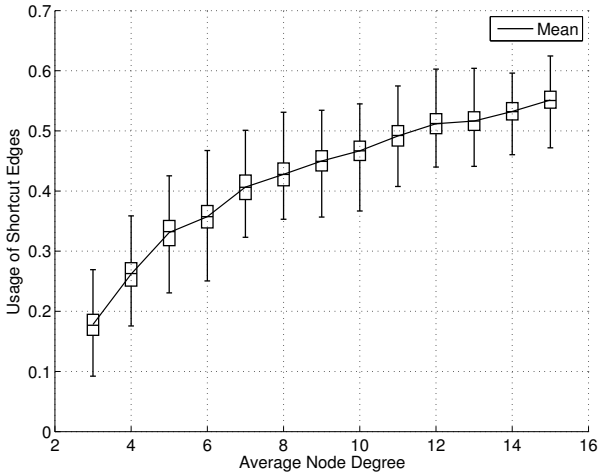

Figure 3. The usage of shortcut edges vs. the average node degree. The boxes show the 25th, 50th and 75th percentile as well as the minimum and the maximum value.

be uniform, and the probability that a shortcut edge incident on S_ℓ is inside Λ_ℓ is approximately $p_\ell = \#\Lambda_\ell/n = d^{-(L-\ell+1)}$ whence the probability of at least one useful shortcut is

$$p_U = 1-(1-p_\ell)^{\delta-d} = 1-\left(1-d^{-(L-\ell+1)}\right)^{\delta-d}$$

for $\ell = 0..L-1$. The key point is that the value of p_U is small for critical average degree values, and additionally, it decreases exponentially as the packet approaches the destination.

Thus we expect that for small node degrees (from the critical interval $[3...8]$), on average the fraction of used shortcuts compared to the total number of hops taken by greedy forwarding will be small. To support this claim numerically, we set up an experiment to determine the usage of non-tree hops. We start by extracting the largest connected component of $G(n_o,p)$ graphs of varying size n_0 and edge probability p, resulting in graphs with $n \approx 50$ nodes and node degrees $[3...15]$. From each such graph we sample 1000 spanning trees uniformly at random and K-embed them. For each such embedding, the usage of shortcuts is recorded. The results are presented in Fig. 3. Indeed, as predicted by our simplified model, small node degrees imply that the greedy routes mainly consist of tree edges. In other words, for node degrees from the critical interval, shortcuts, although present are rarely used for greedy forwarding and this is the reason for the corresponding increase of the average hop stretch.

Based on these observations, we propose a heuristic for choosing spanning trees that yield small hop stretch of greedy graph embeddings. The hop stretch measures the extent to which greedy paths coincide with the shortest paths of the graph. To lower the hop stretch, we need to increase this coincidence. If we choose a spanning tree whose edges represent *as many of the shortest paths in the graph as possible*, the embedding based on this tree will have low hop stretch since the greedily forwarded packets will be taking the tree edges and thus greedy paths will more closely approximate the shortest paths.

To construct a tree consisting of the edges that are most frequently used by the shortest paths in the graph, for experimental purposes, we proceed as follows: We assign to each edge in the graph a weight that represents the total number of shortest paths (for all pairs) that pass through that edge. From this weighted graph we choose the spanning tree of maximum weight and use it as a basis of a greedy embedding. We call this tree the *maximum weight spanning tree* (MWST).

To initially examine the hop stretch properties of K-embeddings based on the MWST, we set up an experiment that correlates the average hop stretch and the utilization of shortcuts in greedy embeddings based the MWST and 1200 spanning trees sampled uniformly at random from the largest connected component of a 60-node $G(n,p)$ graph of average node degree 3.5. Fig. 4 shows the results. For control, the minimum weight spanning tree (mWST) is also included.

We observe from Fig. 4 that for randomly sampled spanning trees, typically only 15–30% of the traversed edges are shortcuts, while the majority of hops (70–85%) taken by greedy forwarding are tree edges for most spanning trees. Some spanning trees provide better utilization of shortcut edges, and as expected this improves the hop stretch of the embedding. Thus, one way to lower the hop stretch appears to be choosing a spanning tree that provides better shortcut utilization. However, this relation holds only on average, and the dependency between the hop stretch and the fraction of shortcuts is weak. On the other hand, the MWST shows an average hop stretch of 1.14 (14%) compared to the values for random trees (20–55%) and at the same time, the MWST renders an embedding having a notably low participation of shortcuts in the greedy paths (about 13%).

To investigate whether the MWST retains the low hop stretch of the embedding for varying node degrees and multiple graph instances, we perform an experiment similar to the one described in Sec. II, but this time including the embeddings based on the MWST as well as the mWST. The results are shown in Fig. 5 and confirm that on average, for the entire interval of critical node degrees, the MWST performs consistently and significantly better than the average random tree and the minimum weight spanning tree. Above the critical interval, the three curves coalesce since more and more nodes are directly connected and greedy forwarding trivially finds the one-hop paths to the destinations.

Further evaluation of the MWST heuristic over a wider class of test cases is presented in Sec. IV.

B. Minimum Diameter Spanning Tree Heuristics

In this section, we investigate the usability of an alternative low-stretch heuristic for K-embedding – the minimum-diameter spanning trees (mDST).

From our previous considerations it appears that a spanning tree "representing" a large number of shortest paths of the graph, provides relatively low hop stretch. Since the maximum path length in a minimum diameter spanning tree of the graph is the lowest among the spanning trees, we conjecture that mDSTs will, like MWSTs, also represent a large number of

Figure 4. The average hop stretch vs. the fraction of shortcut hops taken by all greedy routes for a random, 60-node graph. The fraction of shortcuts (non-tree edges) used by the greedy paths is relatively small. (The line fits the data best in the least-squares sense.)

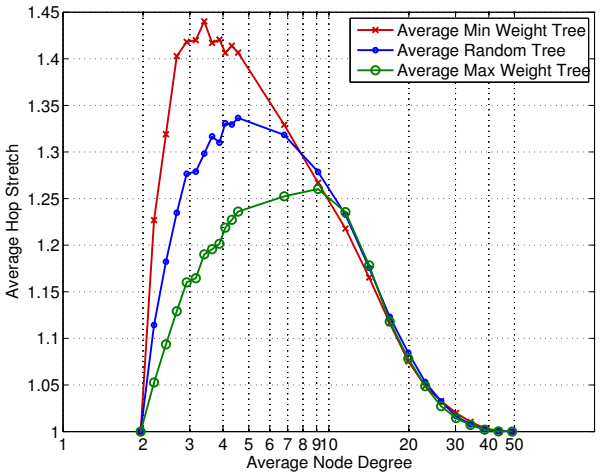

Figure 5. Dependence of the average hop stretch of a greedy embedding on the average node degree of the graph, averaged over 32 instance of a $G(n_0, p)$ graph. There is a range of critical node degrees, typically between 3 and 8 for which the hop stretch is maximal.

shortest paths and thus provide low hop stretch compared to a randomly sampled spanning tree.

As a first experiment, we examine the correlation between the spanning tree diameter and its weight as defined in Sec. III. For this purpose, we sample 2000 spanning trees uniformly at random from a 50-node graph representing the largest connected component of a $G(n, p)$ graph with an average node degree of 4. Typical results are illustrated in Fig. 6. along with the MWST, mWST, and mDST. We observe that while the mDST does not capture as much graph weight as the MWST, it still has more weight than most random spanning trees. This encourages further investigation of the hop stretch provided by mDSTs.

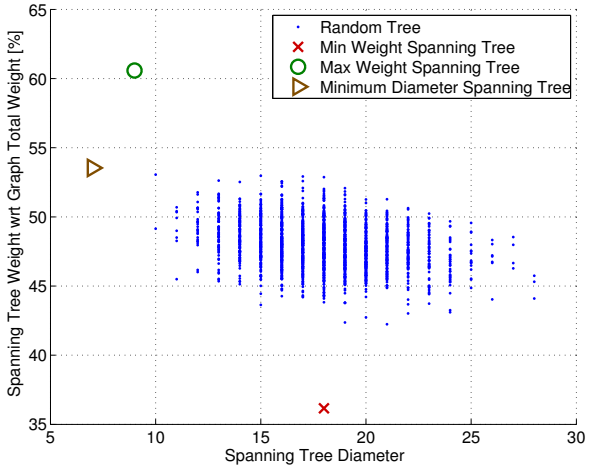

Figure 6. Spanning tree weight percentage of the total graph weight vs. spanning tree diameter for 2000 spanning trees sampled uniformly at random and used as a basis for the greedy embedding of a random 50-node graph. The MWST, mWST and mDST are also shown.

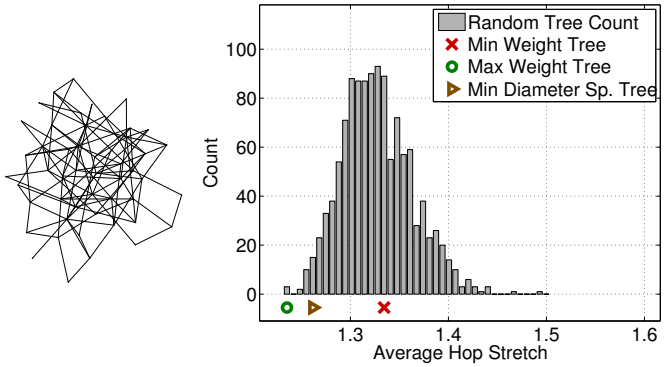

Figure 7. Typical hop stretch results for a $G(n, p)$ graph

IV. Empirical Evaluation

In this section we present additional comparative results on various properties of the MWST and mDST heuristic.

So far we have considered random graph instances generated from the $G(n, p)$ model. Here we also use graphs generated by placing nodes uniformly at random in the Euclidean plane and placing an edge between pairs of nodes if the Euclidean distance between them is below a chosen threshold. We refer to this model as a *wireless graph.* The average node degree of the wireless graph can be varied by varying the threshold distance. We use the largest connected component from the generated graphs.

Figures 7 and 8 illustrate instances of connected components of $G(n, p)$ and wireless graphs, as well as typical histograms of the average hop stretch for 400 spanning trees sampled uniformly at random. The MWST, mWST and mDST are also shown.

Figures 9 and 10 show the correlation between the average hop stretch and the spanning tree diameter for the $G(n, p)$ and the wireless graph model. We observe that for the general random spanning tree, there is no strong correlation between

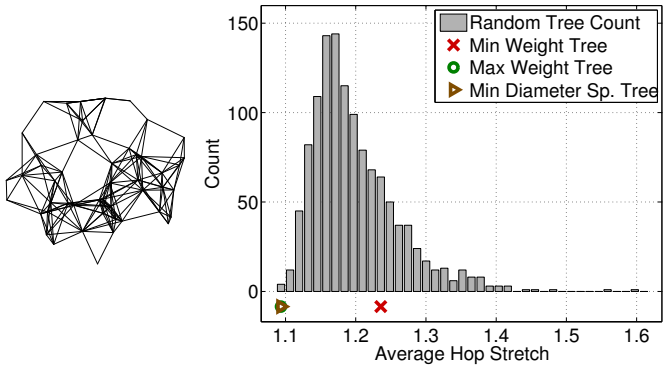

Figure 8. Typical hop stretch results for a wireless graph

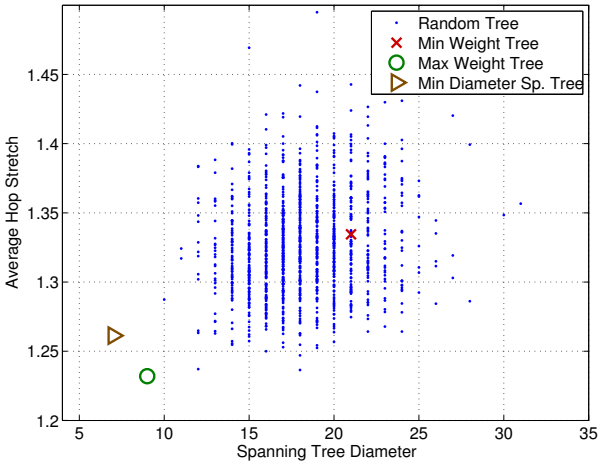

Figure 9. Average hop stretch vs. spanning tree diameter for the $G(n,p)$ graph model

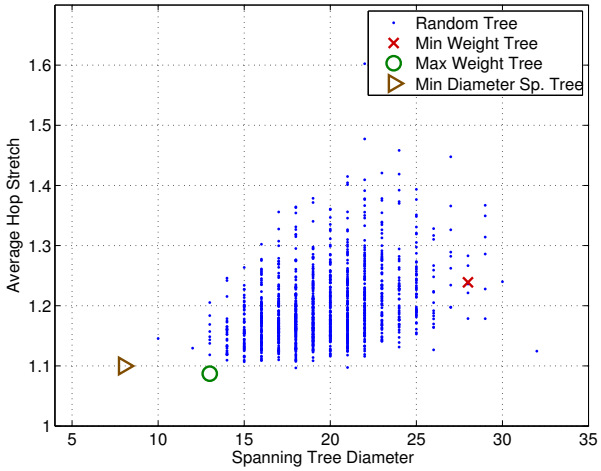

Figure 10. Average hop stretch vs. spanning tree diameter for the wireless graph model

the two quantities, but the correlation becomes notable toward the low stretch - low diameter end of the spectrum. We also observe that a spanning tree of maximum weight does not necessarily have the minimum diameter, but the diameter of the MWST is usually low.

We conclude that for both graph types, MWST consistently shows the best stretch performance. The mDST has somewhat higher, but still satisfactorily low hop stretch.

V. Implementation Aspects

Typically, in an unweighted network graph, there are multiple shortest paths for a pair of nodes. In generating edge weights for the purpose of creating the MWST, we took into consideration all possible shortest paths in our graphs. To calculate all shortest paths, for each pair, we used a slightly modified version of an efficient algorithm for finding the k-shortest loopless paths in a network [14]. Once the weighted graph is obtained from the initial unweighted topology, finding the MWST or the mWST is a matter of applying a classical minimum spanning tree algorithm (e.g. [15], [16]).

The mDST heuristic, on the other hand, seems more amenable for implementation in that finding minimum diameter spanning tree can be performed efficiently ([17], [18]) and via distributed computation ([19]).

VI. Conclusion

Greedy embeddings [4] are a promising tool for information routing in networks, including electronic and social networks. The advantages of greedy embedding are that message routing is guaranteed to succeed, even though nodes maintain information only about their directly connected neighbors. Greedy embedding is a hard problem, but recently several interesting solutions have been proposed [8], [9], [10], [11]. However, a problem left open in these works has been to find greedy embeddings that are low-stretch – meaning that the paths taken in such networks have length close to that of the shortest path.

In this work we studied how topological and geometric properties of greedy embeddings [8] influence the hop stretch. Based on the obtained insights, we constructed the maximum-weight spanning tree (MWST) and inferred the minimum-diameter spanning tree (mDST) heuristics for reduction of hop stretch. We provide arguments and insights suggesting why our proposed heuristics are justified. We note that our heuristics can be applied to any greedy embedding procedure based on the extraction of an arbitrary spanning tree, including those described in [8], [9], [10], [11]. For the graph models considered in our evaluations, these heuristics typically improve average hop stretch from e.g. 50% (worst) or 30% (average) to <15%. The MWST is the best performer most of the time. The mDST heuristic is easier to calculate in a distributed fashion.

Acknowledgments

This work was supported by the National Science Foundation under Grant no. CNS-1018266.

REFERENCES

[1] G. G. Finn, "Routing and addressing problems in large metropolitan-scale internetworks," Tech. Rep. ISI/RR-87-180, Univ. of Southern California, Information Sciences Institute, March 1987.

[2] H. Takagi and L. Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals," *IEEE Transactions on Communications*, vol. 32, no. 3, pp. 246–257, 1984.

[3] R. Nelson and L. Kleinrock, "The spatial capacity of a slotted aloha multihop packet radio network with capture," *IEEE Transactions on Communications*, vol. 32, pp. 684–694, Jun 1984.

[4] C. H. Papadimitriou and D. Ratajczak, "On a conjecture related to geometric routing," *Theoretical Computer Science*, vol. 344, no. 1, pp. 3 – 14, 2005.

[5] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, MobiCom '03, (New York, NY, USA), pp. 96–108, ACM, 2003.

[6] M. Mauve, A. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *Network, IEEE*, vol. 15, pp. 30–39, Nov/Dec 2001.

[7] S. Giordano, I. Stojmenovic, and L. Blazevic, "Position based routing algorithms for ad hoc networks: a taxonomy," in *Ad Hoc Wireless Networking*, pp. 103–136, Kluwer, 2003.

[8] R. Kleinberg, "Geographic routing using hyperbolic space," in *Proceedings of IEEE Infocom 2007*, pp. 1902–1909, May 2007.

[9] A. Cvetkovski and M. Crovella, "Hyperbolic embedding and routing for dynamic graphs," in *Proceedings of IEEE Infocom 2009*, Apr. 2009.

[10] D. Eppstein, "Manhattan orbifolds," *Topology and its Applications*, vol. 157, no. 2, pp. 494 – 507, 2010.

[11] D. Eppstein and M. T. Goodrich, "Succinct greedy geometric routing using hyperbolic geometry," *IEEE Transactions on Computers*, vol. 60, pp. 1571–1580, 2011.

[12] J. W. Anderson, *Hyperbolic Geometry*. Springer, 2nd ed., 2007.

[13] D. J. Aldous, "The random walk construction of uniform spanning trees and uniform labelled trees," *SIAM J. Discret. Math.*, vol. 3, pp. 450–465, November 1990.

[14] J. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[15] J. Kruskal Jr., "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, pp. 48–50, 1956.

[16] R. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, vol. 36, pp. 1389–1401, 1957.

[17] P. Camerini, G. Galbiati, and F. Maffioli, "Complexity of spanning tree problems: Part I," *European Journal of Operational Research*, vol. 5, no. 5, pp. 346 – 352, 1980.

[18] R. Hassin and A. Tamir, "On the minimum diameter spanning tree problem," *Information Processing Letters*, vol. 53, no. 2, pp. 109 – 111, 1995.

[19] M. Bui, F. Butelle, and C. Lavault, "A distributed algorithm for constructing a minimum diameter spanning tree," *Journal of Parallel and Distributed Computing*, vol. 64, no. 5, pp. 571–577, 2004.