Netw Sci (2013) 3: 2–12
DOI 10.1007/s13119-013-0013-7

RESEARCH ARTICLE

On the choice of a spanning tree for greedy embedding of network graphs

Andrej Cvetkovski (✉), **Mark Crovella**

Department of Computer Science, Boston University, 111 Cummington Mall, Boston, MA 02215, USA

Received: 30 September 2012/Revised: 19 December 2012/Accepted: 8 January 2013

Abstract Greedy embedding is a graph embedding that makes the simple greedy geometric packet forwarding successful for every source-destination pair. It is desirable that graph embeddings also yield low hop overhead (stretch) of the greedy paths over the corresponding 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 design embedding heuristics that yield minimal hop stretch greedy embeddings and verify their effectiveness on models of synthetic graphs. Finally, we verify the effectiveness of our heuristics on instances of several classes of large, real-world network graphs.

Keywords greedy routing, shortest path, hyperbolic geometry, network graph, AS graph, hop stretch

1 Introduction

Greedy routing as an efficient alternative to the classical routing table approaches, was studied initially in [9,24,19]. In the greedy routing paradigm, the communication network is first embedded in a metric space by assigning to each node a coordinate denoting its location. From these coordinates, one can calculate the geometric distance between any two nodes in the embedding. According to the simplest deterministic greedy routing rule, a node forwards a message to a directly connected neighboring node that is closest to the destination in geometric distance.

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 only on the coordinates of its neighbors. In addition, greedy routing based on node locations and distances in Euclidean space, as can be demonstrated by simulations, has high success rate[1]. However, a principal disadvantage of greedy routing is that it is not guaranteed to succeed for every source-destination pair: it is easy to construct examples where a packet reaches a node that is closer to the destination than all of its direct neighbors, and the forwarding fails even though a path to the destination exists (see e.g. [15] for more details).

Routes found by greedy forwarding may 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 and the corresponding shortest path in the graph. The *average* and the *maximum hop stretch* can be used as measures of the hop stretch for the entire graph. Ideally, the greedy path and the shortest path would coincide for each pair, in which case the hop stretch is 1. In practice, however, the hop stretch is often higher, implying reduced efficiency of the routing. Minimizing the hop stretch of greedy routing is thus an important problem and has received significant attention in the research literature.

Both the rate of success and the hop stretch of the forwarding depend on the graph topology but also on the embedding. 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* [20].

If the real-world (geographic) node locations are used as coordinates for forwarding, the success rate and hop

E-mail: andrej@bu.edu

A preliminary version of this work appears in the Fourth International Workshop on Network Science for Communication Networks (NetSciCom'12) [6].

[1] The rate of success of greedy routing for a given graph embedding is the fraction (percentage) of all source-destination pairs for which greedy routing succeeds in finding the path to the destination.

stretch can not be influenced readily. This motivates the study of network embeddings based on virtual coordinates (not necessarily coinciding with the physical node locations) [22], chosen so as to optimize success rate, hop stretch or some other quantity of interest (see also e.g. [17,10] and the references therein). Limiting the focus to metric spaces and virtual coordinates, [12] presents a notable result in this direction: 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 low hop stretch. Using virtual coordinates, we employ the procedures of Kleinberg [12] and Cvetkovski and Crovella [5] to obtain greedy embeddings. (Henceforth, we refer to the greedy embeddings [12] and [5] as *K embedding* and *C embedding* respectively.) Motivated by the observation that varying some of the parameters of a greedy embedding can cause significant changes to the obtained average and maximum hop stretch, we start by studying how the properties of K-embedded graphs influence the hop stretch. Our finding is that the choice of a spanning tree is a central problem in the reduction of the hop stretch of greedy embeddings based on spanning trees. We use the obtained insights to design embedding heuristics that yield low hop stretch greedy embeddings. Subsequently, we verify the effectiveness of our proposed heuristics on a range of synthetic and real-world graphs, using both K and C greedy embeddings.

We emphasize that our heuristics are not limited in applicability to the K or C embeddings and are expected to produce good results with *any* greedy embedding procedure that is based on the extraction of an arbitrary spanning tree, including those described in [12,5,7,8,18], and those that might be devised in the future. To the best of our knowledge, this is the first study to demonstrate these insights and to construct corresponding heuristics.

The rest of this paper is organized as follows. Section 2 provides more specific background on the problem at hand. We present the main ideas of our approach in Section 3 and further examine their qualities in Section 4. The conclusions are in Section 5.

2 Preliminaries

The input to the K embedding algorithm [12] is a finite, connected, undirected and unweighted graph representing a communication network. The procedure places the graph nodes in the hyperbolic plane, and uses the standard hyperbolic distance (see e.g. [2]) for the forwarding function. The generic algorithm of [12] finds a greedy embedding of the infinite d-regular tree for any integer $d \geqslant 3$. 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. It is proved in [12] 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 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 $G(n_0, p)$ 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 [1], and average the hop stretch over 32 graph instances. The results are shown in Figure 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].

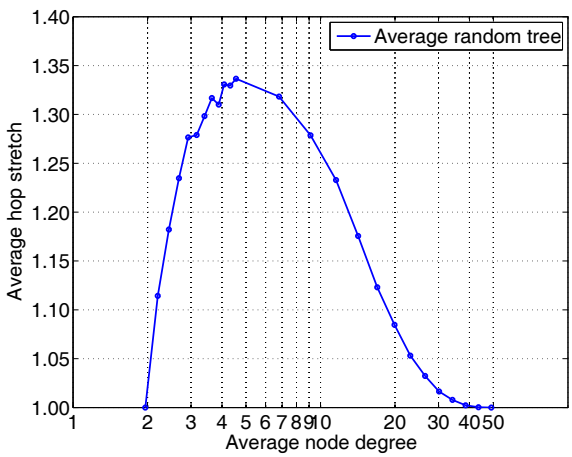


Fig. 1 Dependence of the average hop stretch of a greedy embedding on the average node degree of the graph, averaged over 32 instances of a $G(n_0, p)$ graph. There is a range of critical node degrees, typically between 3 and 8 for which the hop stretch is maximal

While the hop stretch depends both on 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 using the K embedding 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.

3 Heuristics

3.1 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 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, Figure 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 $\overline{\delta} = 6$ leaves on average 2 non-tree edges incident on each node

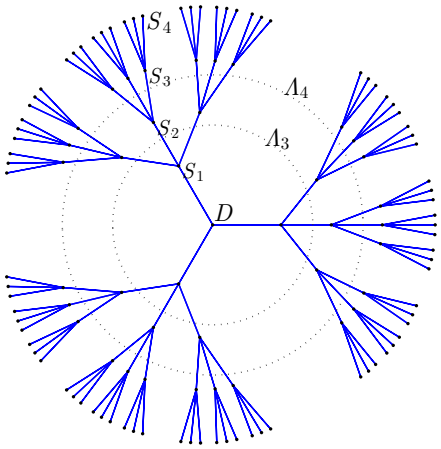


Fig. 2 A simplified model: the number of next hop candidates exponentially decreases as the packet gets closer to the destination

(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 $\#\Lambda_\ell = n/(d-1)^{L-\ell+1}$ nodes. The probability of a non-tree edge between S_ℓ and any other node is assumed to be uniform, and the probability that a shortcut edge incident on S_ℓ is inside Λ_ℓ is approximately $p_\ell = \#\Lambda_\ell/n = (d-1)^{-(L-\ell+1)}$ whence the probability of at least one useful shortcut is

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

for $\ell = 2\,..\,L-1$. 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 $G(n_0, p)$ 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 Figure 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

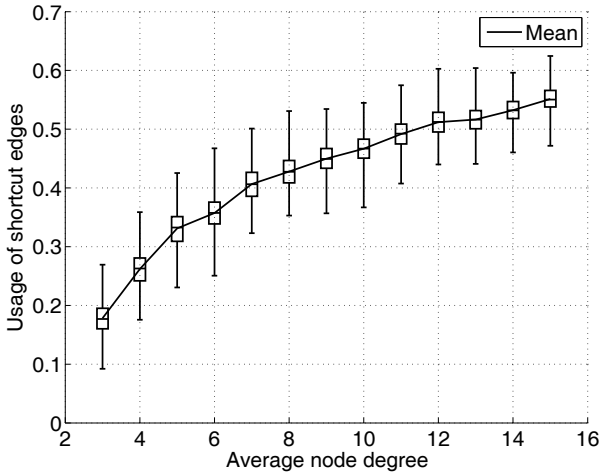


Fig. 3 The usage of shortcut edges vs. the average node degree. The boxes show the 25th, 50th and 75th percentile as well as the minimum and the maximum value

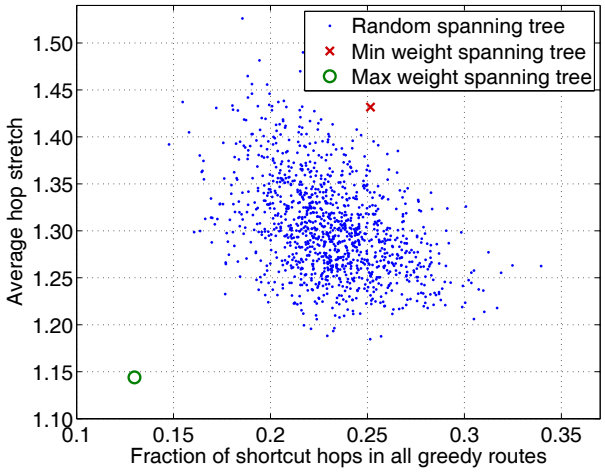


Fig. 4 The average hop stretch vs. the fraction of shortcut hops taken by all greedy routes for a random, 60-node graph. The fraction of shortcuts (non-tree edges) used by the greedy paths is relatively small

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 on 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. Figure 4 shows the results. For control, the minimum weight spanning tree (mWST) is also included.

We observe in Figure 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 Section 2, but this time including the embeddings based on the MWST as well as the mWST. The results are shown in Figure 5 and confirm that on average,

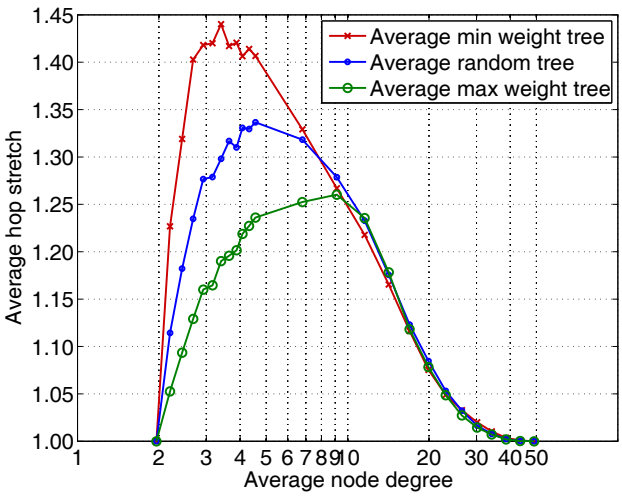


Fig. 5 Dependence of the average hop stretch of a greedy embedding on the average node degree of the graph, averaged over 32 instances of a $G(n_0, p)$ graph. There is a range of critical node degrees, typically between 3 and 8 for which the hop stretch is maximal

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 Section 4.

3.2 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 short paths of the graph, provides relatively low hop stretch. Since the minimum diameter spanning tree minimizes the maximum path length on the tree among all the spanning trees, we conjecture that mDSTs will, like MWSTs, also represent a large number of short 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 Section 3. 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 Figure 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.

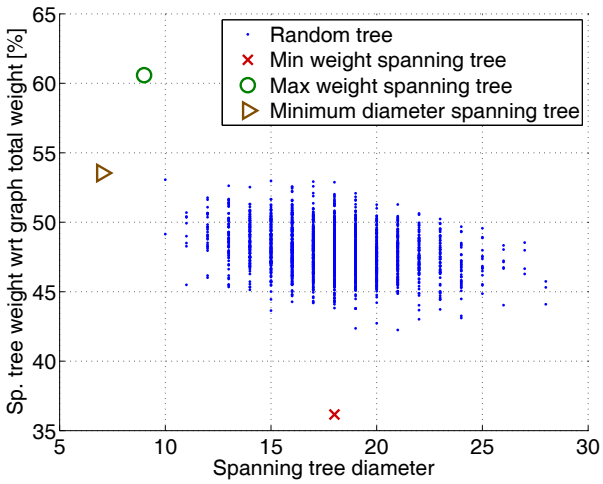


Fig. 6 Spanning tree weight percentage of the total graph weight vs. spanning tree diameter for 2000 spanning trees sampled uniformly at random and used as a basis for the greedy embedding of a random 50-node graph. The MWST, mWST and mDST are also shown

4 Numerical evaluation

4.1 Numerical evaluation for synthetic graphs

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 of 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 as well as the 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 the two quantities, but the correlation becomes notable toward the low stretch—low diameter end of

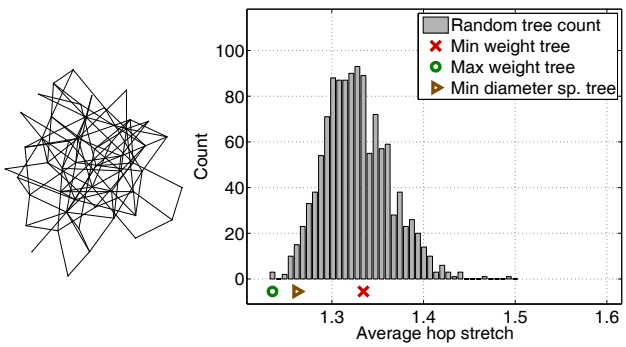


Fig. 7 Typical hop stretch results for a $G(n, p)$ graph

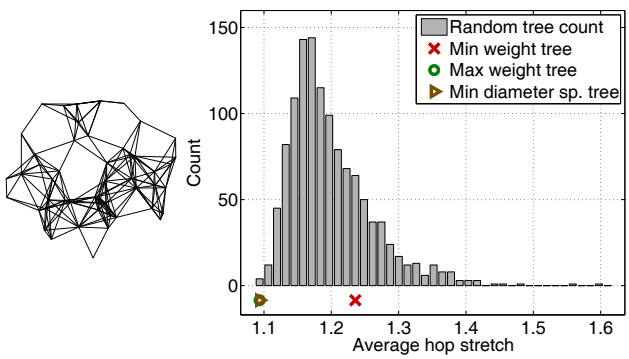


Fig. 8 Typical hop stretch results for a wireless graph

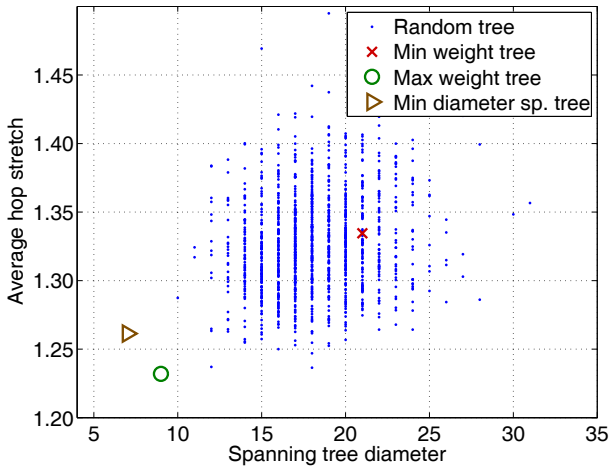


Fig. 9 Average hop stretch vs. spanning tree diameter for the $G(n, p)$ graph model

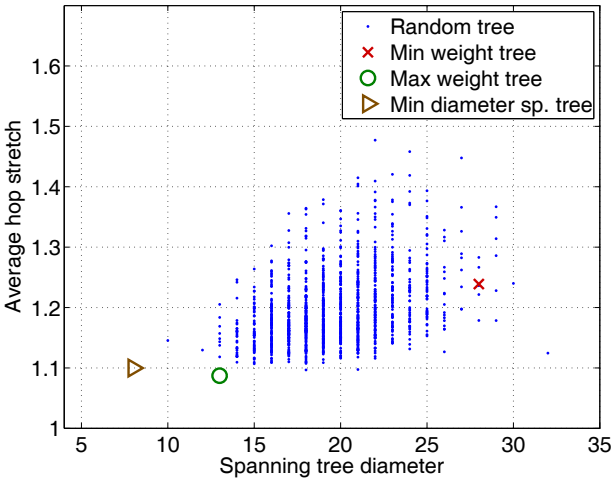


Fig. 10 Average hop stretch vs. spanning tree diameter for the wireless graph model

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.

4.2 Numerical evaluation for large real-world graphs

Prompted by the potentially high computational cost of the MWST heuristic as described in Section 3.1, in this section we study the performance of MWSTs computed from a relatively small sample of all pairs and a single sample from the possibly many shortest paths between each sampled pair. Henceforth, for brevity, we call such trees *sampled MWSTs*. Sampled MWSTs allow for significant reduction of computation costs for large networks consisting of thousands of nodes and thus, billions of source-destination pairs. Moreover, sampled MWSTs allow for a design trade-off between performance and cost. Here we demonstrate that just like full MWSTs, sampled MWSTs also perform consistently and significantly better than the average random spanning tree. Using sampled MWSTs, we then proceed to show the applicability of the MWST heuristic on instances of several classes of large real-world network graphs.

4.2.1 *Sampling*

To get an initial sense of how the performance of sampled MWSTs and mWSTs depends on the number of sampled pairs used for the calculation of the weighted graph, we set up the following experiment: We sample pairs of nodes uniformly at random from the graph to be embedded, and for each such pair we calculate one shortest path between them, also chosen uniformly at random from all the possible shortest paths for that pair. Starting from the adjacency matrix of the graph, we increment by 1 those entries that correspond to an edge of the calculated shortest path. The MWST as well as the mWST are subsequently calculated from the so obtained sampled weight matrix and for the corresponding K embeddings, the average and the maximum hop stretch are evaluated for greedy routing between all pairs.

The hop stretch results are shown in Figure 11 for the GCC of a $G(n, p)$ graph consisting of 100 nodes (10 000 pairs). The graph is sampled progressively from 0 to 10% of all possible pairs and hop stretch is calculated for several data points from that interval. We observe that for a fraction of sampled pairs as small as 2%, the worst case of the average stretch of the MWST in 150 experiment runs is better than the best case of the mWST tree. We also observe that increasing the number of samples above this point soon starts to provide diminishing returns.

4.2.2 *Real-world graphs*

Using sampled spanning trees we are in a position to apply the MWST heuristic to large real-world network graphs. For our empirical evaluation, we choose the four network instances summarized in Table 1 and detailed below.

HEP–PH [16] is a collaboration network containing edges between authors who co-authored at least one paper in the High-Energy Physics–Phenomenology category of the ArXiv preprint database between January 1993 and April 2003 (124 months). It begins a few months after ArXiv's inception and thus represents a fairly complete history of the HEP–PH section up to 2003. A paper co-authored by k authors induces a completely connected subgraph on k nodes.

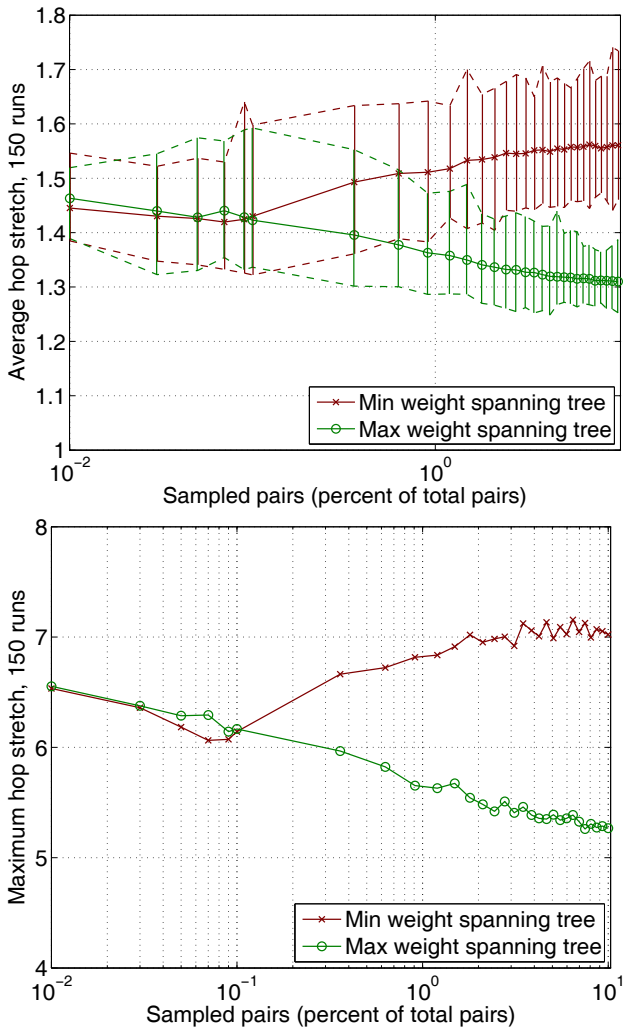


Fig. 11 The average and maximum hop stretch as a function of the percent of sampled pairs for the calculation of the weight matrix. The curves are averaged over 150 runs of the experiment. The vertical lines show the min-max range for each data point. For a fraction of pairs as small as 2%, the worst case of the MWST was better than the best case of the mWST tree

Table 1 Large, real-world network graphs used in the evaluation of the MWST heuristics. Number of nodes in the GCC n_{GCC}; average node degree $\overline{d}$

Graph	n_{GCC}	$\overline{d}$	Type
ArXiv HEP–PH [16]	11204	20.996	Collaboration
Enron Email [16,13]	33696	10.732	Communication
Gnutella [16,23]	22663	4.827	Internet P2P
AS Graph [26]	41491	6.282	Internet AS

The Enron Email Network [13] is a communication network describing email communication of Enron, a US based energy, commodities and services company. Nodes in the network are email addresses and there is an edge if an address i sent at least one email to the address j. Here we use the graph described in [16], containing data from October 2003 to May 2005.

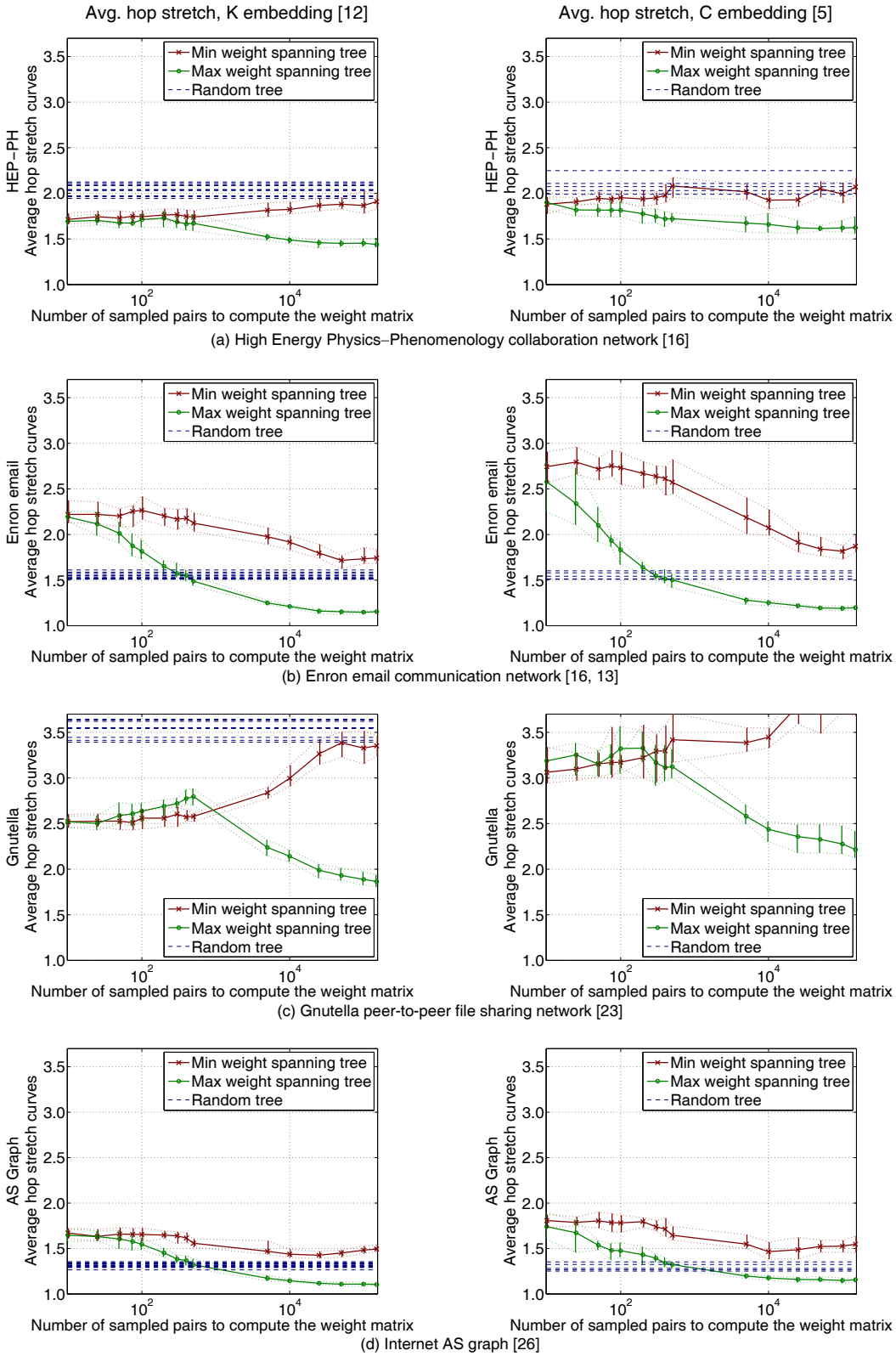
Gnutella [23] is peer-to-peer file sharing network and represents an example of a large self-organized network of independent entities. We use a snapshot from 03/25/2002 [16]. Nodes represent the hosts in the Gnutella network and edges represent file transfers between them.

The AS (Autonomous System) Graph is a graph representing the topology of the Internet at the inter-domain level where each AS is a node, and the BGP peering between two ASes is a link. We use a snapshot of the AS Graph from the Internet Topology Collection Project at UCLA [26] as of 04/24/2012 and extract the links that have been seen in the last 20 days.

We set up an experiment to evaluate the performance of the MWST heuristic on each of these four large networks, the results of which are presented in Figures 12 and 13. For each network, we start by sampling progressively from 10 to 150 000 pairs and the sampled weighted graph is calculated for several data points from that interval. For each data point, we find the MWST and the mWST and calculate the corresponding K and C greedy embeddings. Then we find the length of the greedy path for 300 pairs chosen uniformly at random from the network. In Figures 12 and 13 we report the average and maximum hop stretch for 10 runs of this experiment. The maximum, minimum and the average of the series are depicted. In order to demonstrate that MWST consistently performs better than a randomly chosen spanning tree, for each graph we also report the hop stretch for 10 spanning trees chosen uniformly at random from the graphs.

In all cases, we observe that even for a very small fraction of pairs there is a significant improvement of the hop stretch when using the MWST compared to a random spanning tree. In the Enron dataset, MWST calculated from as few as 1000 pairs provided better results than a random tree in all 10 runs of the experiment. In the AS Graph, 4000 random pairs were sufficient to see a definite improvement. In the cases of HEP–PH and Gnutella, MWST was better than a random tree for any number of samples and notably improved with increasing of the number of samples above 5000. In all cases, saturation in the improvement for both the average and the maximum stretches appeared above 50 000 pairs which is 0.04%, 0.004%, 0.01%, and 0.003% of all pairs for the HEP–PH, Enron, Gnutella and the AS Graph, respectively.

The average hop stretch for 100 000 samples is summarized in Table 2.



(a) High Energy Physics–Phenomenology collaboration network [16]

(b) Enron email communication network [16, 13]

(c) Gnutella peer-to-peer file sharing network [23]

(d) Internet AS graph [26]

Fig. 12 Numerical evaluation **of the average hop stretch** on large real-world network graphs. The left column shows the average hop stretch for 10 sampled weight matrices and 300 sampled source-destination pairs for the **K embedding** [12]. The right column shows the average hop stretch for the same cases when the **C embedding** [5] is used. The vertical bars show the minimum and the maximum values among the experiments. The hop stretch for 10 random spanning trees is also shown. The corresponding axes and data points are identical. Note that in each case the MWST performed significantly better than a randomly chosen spanning tree in both the average and the maximum stretch

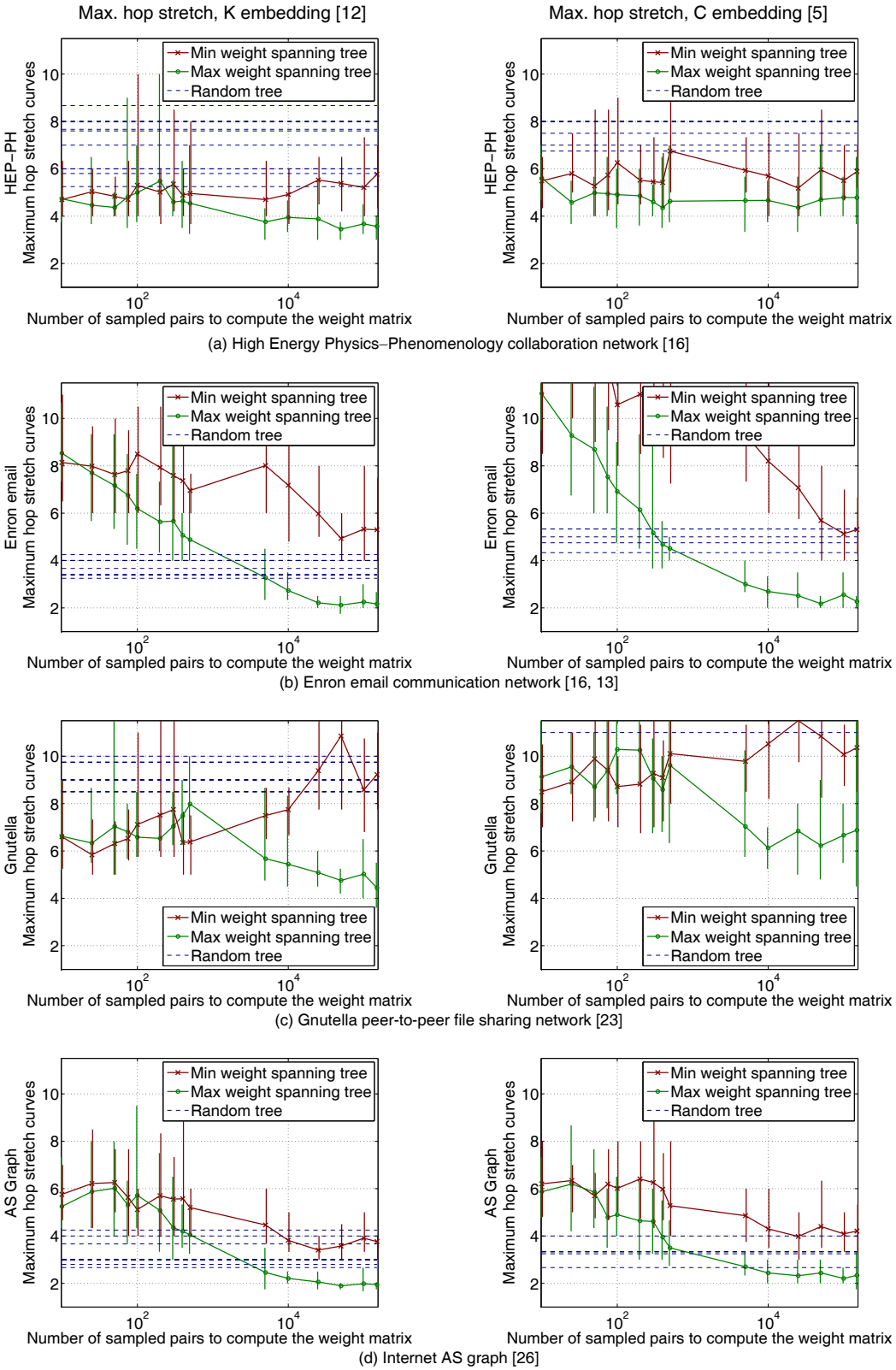


Fig. 13 Numerical evaluation **of the maximum hop stretch** on large real-world network graphs. The left column shows the average of the maximum hop stretch for 10 sampled weight matrices and 300 sampled source-destination pairs for the **K embedding** [12]. The right column shows the average of the maximum hop stretch for the same cases when the **C embedding** [5] is used. The vertical bars show the minimum and the maximum values among the experiments. The hop stretch for 10 random spanning trees is also shown. The corresponding axes and data points are identical. Note that in each case the MWST performed significantly better than a randomly chosen spanning tree in both the average and the maximum stretch

Table 2 Average hop stretch at 100 000 sampled pairs, K embedding

Graph	MWST	Random	mWST
ArXiv HEP–PH	1.45	2.04 (+0.59)	1.87 (+0.42)
Enron Email	1.15	1.55 (+0.40)	1.73 (+0.58)
Gnutella	1.89	3.53 (+1.64)	3.33 (+1.44)
AS Graph	1.11	1.32 (+0.21)	1.48 (+0.37)

4.3 Implementation details

In this section we provide additional details on the implementation of our low-stretch heuristics.

In a network graph there may be multiple shortest paths between a pair of nodes. In generating edge weights for the purpose of validating the MWST on synthetic graphs, we took into consideration all possible shortest paths between each pair. 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 [25]. 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 weight spanning tree algorithm (e.g. [14,21]). However, in the evaluation of the MWST heuristic on large real-world networks (Section 4.2), we showed that using only one randomly chosen shortest path also provides the desired reduction of the hop stretch. Given that the input graph is unweighted, shortest paths can be found by a randomized breadth-first search from one source to many or all destinations with time complexity of $O(N+E)$, where N and E are the number of nodes and edges respectively. On the other hand, a minimum diameter spanning tree [4,11] can be performed in linear time via distributed asynchronous computation [3]. Finally, we showed that sampling can be used to greatly reduce the computational cost of the weight matrix for real-world networks.

5 Conclusions

Greedy embeddings are a promising tool for information routing in communication and social networks. The main advantage of greedy embedding is that packet forwarding is guaranteed to succeed while nodes maintain information only about their directly connected neighbors. Greedy embedding is a hard problem, and recently several interesting solutions have been proposed [12,5,7,8,18]. 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 study how topological and geometric properties of greedy embeddings influence the hop stretch. Our finding is that the choice of a spanning tree is a central problem in the reduction the hop stretch of greedy embeddings based on spanning trees. From the obtained insights of our study, we construct the maximum-weight spanning tree (MWST) heuristic from which we derive the minimum-diameter spanning tree (mDST) and the sampled MWST heuristics for reduction of the hop stretch. We provide arguments and insights explaining why our proposed heuristics are justified.

We emphasize that our heuristics can be applied to any greedy embedding procedure based on the extraction of an arbitrary spanning tree, including those described in [12,5,7,8,18]. Furthermore, a spanning tree constructed by any method that gathers sufficient weight according to our definition is likely to perform well. This observation opens the problem of constructing efficient methods for the (preferably distributed) calculation of high-weight spanning trees for low-stretch greedy embeddings.

For the graph models considered in our evaluations, these heuristics typically improve average hop stretch from e.g. 1.50 (worst) or 1.30 (average) to <1.15 in the case of $G(n, p)$ or wireless graphs. For the considered instances of large real-world graphs we measured a reduction of the average hop stretch between 0.21 and 1.64 with respect to a randomly chosen spanning tree, for a modest fraction of sampled pairs for the calculation of the MWST. Overall, the MWST is the best performer most of the time. The derived heuristics (mDST and sampled MWST) appear more amenable for implementation in real-world graphs and perform just as well as the full-fledged MWST.

Acknowledgements

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

References

[1] D. J. Aldous, "The random walk construction of uniform spanning trees and uniform labelled trees," *SIAM J. Discrete Math.*, vol. 3, no. 4, pp. 450 – 465, Nov. 1990.

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

[3] M. Bui, F. Butelle, and C. Lavault, "A distributed algorithm for constructing a minimum diameter spanning tree," *J. Parallel Distrib. Comput.*, vol. 64, no. 5, pp. 571 – 577, May 2004.

[4] P. M. Camerini, G. Galbiati, and F. Maffioli, "Complexity of spanning tree problems: Part I," *Eur. J. Oper. Res*, vol. 5, no. 5, pp. 346 – 352, Nov. 1980.

[5] A. Cvetkovski and M. Crovella, "Hyperbolic embedding and routing for dynamic graphs," in *Proc. IEEE Infocom 2009*, pp. 1647 – 1655.

[6] A. Cvetkovski and M. Crovella, "Low-stretch greedy embedding heuristics," in *4th Int. Workshop Network Science for*

Communication Networks (NetSciCom'12), 2012, pp. 232 – 237. Orlando, USA.

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

[8] D. Eppstein and M. T. Goodrich, "Succinct greedy geometric routing using hyperbolic geometry," *IEEE Trans. Comput.*, vol. 60, no. 11, pp. 1571 – 1580, Nov. 2011.

[9] G. G. Finn, "Routing and addressing problems in large metropolitan-scale internetworks," Technical Report ISI/RR-87-180, University of Southern California, Information Sciences Institute, 1987.

[10] S. Giordano, I. Stojmenovic, and L. Blazevic, "Position based routing algorithms for ad hoc networks: A taxonomy," in *Ad HocWireless Networking*, X. Chen, X. Huang, and D. Du, Eds. Boston: Kluwer, 2003, pp. 103 – 136.

[11] R. Hassin and A. Tamir, "On the minimum diameter spanning tree problem," *Inform. Process. Lett.*, vol. 53, no. 2, pp. 109 – 111. Jan. 1995.

[12] R. Kleinberg, "Geographic routing using hyperbolic space," in *Proc. IEEE Infocom 2007*, pp. 1902 – 1909.

[13] B. Klimt and Y. Yang, "Introducing the Enron corpus," presented at the 1st Conf. Email and Anti-Spam (CEAS), Mountain View, CA, USA, 2004.

[14] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proc. Amer. Math. Soc*, vol. 7, no. 1, pp. 48 – 50, Feb. 1956.

[15] B. Leong, "New techniques for geographic routing," Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, USA, 2006.

[16] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, Article no. 2, Mar. 2007.

[17] M. Mauve, J. Widmer, and H. Hartenstein, "A survey on position-based routing in mobile ad hoc networks," *IEEE Network*, vol. 15, no. 6, pp. 30 – 39, Nov. – Dec. 2001.

[18] P. Maymounkov, "Greedy embeddings, trees, and Euclidean vs. Lobachevsky geometry," [online]. Available: http://pdos.csail.mit.edu/~petar/papers/maymounkov-greedy-prelim.pdf

[19] R. Nelson and L. Kleinrock, "The spatial capacity of a slotted aloha multihop packet radio network with capture," *IEEE Trans. Commun.*, vol. 32, no. 6, pp. 684 – 694, Jun. 1984.

[20] C. H. Papadimitriou and D. Ratajczak, "On a conjecture related to geometric routing," *Theor. Comput. Sci.*, vol. 344, no. 1, pp. 3 – 14, Nov. 2005.

[21] R. C. Prim, "Shortest connection networks and some generalizations," *Bell System Tech. J.*, vol. 36, no. 6, pp. 1389 – 1401, Nov. 1957.

[22] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *Proc. 9th Annu. Int. Conf. Mobile Computing and Networking, MobiCom' 03*, New York: ACM, 2003, pp. 96 – 108.

[23] M. Ripeanu, A. Iamnitchi, and I. Foster, "Mapping the Gnutella network," *IEEE Internet Comput.*, Vol. 6, no. 1, pp. 50 – 57, Jan. 2002.

[24] H. Takagi and L. Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals," *IEEE Trans. Commun.*, vol. 32, no. 3, pp. 246 – 257, Mar. 1984.

[25] J. Y. Yen, "Finding the K shortest loopless paths in a network," *Manage. Sci.*, vol. 17, no. 11, pp. 712 – 716, Jul. 1971.

[26] B. Zhang, R. Liu, D. Massey, and L. Zhang, "Collecting the internet as-level topology," *ACM SIGCOMM Comput. Commun. Rev.* vol. 35, no. 1, pp. 53 – 61, Jan. 2005.