

Measurement Manipulation and Space Selection in Network Coordinates

Cristian Lumezanu and Neil Spring
University of Maryland
{lume,nspring}@cs.umd.edu

Abstract

*Internet coordinate systems have emerged as an efficient method to estimate the latency between pairs of nodes without any communication between them. However, most coordinate systems have been evaluated solely on data sets built by their authors from measurements gathered over large periods of time. Although they show good prediction results, it is unclear whether the accuracy is the result of the system design properties or is more connected to the characteristics of the data sets. In this paper, we revisit a simple question: how do the features of the embedding space and the inherent attributes of the data sets interact in producing good embeddings? We adapt the Vivaldi algorithm to use Hyperbolic space for embedding and evaluate both Euclidean and Hyperbolic Vivaldi on seven sets of real-world latencies. Our results show that node filtering and latency distributions can significantly influence the accuracy of the predictions. For example, although Euclidean Vivaldi performs well on data sets that were chosen, constructed and filtered by the designers of the algorithm, its performance and robustness decrease considerably when run on third-party data sets that were not filtered a priori. Our results offer important insight into designing and building coordinate systems that are both **robust** and **accurate** in Internet-like environments.*

1 Introduction

Internet coordinate systems estimate the latency between pairs of nodes without any communication between them. They associate each node to a position in a finite coordinate space; the latency between two nodes is then estimated as the distance between their positions in the space [15, 17, 26, 11, 14, 20, 3, 4]. Coordinate systems are an effective building block for distributed systems and protocols that allow the choice of communication peers based on the latency metric. For example, in the construction of structured overlay networks [23, 19, 18], nodes usually select the closest peers to optimize queries and responses. In file-sharing applications [6, 2] or content distribution networks [5], it is more efficient to download the required content from the server with the lowest latency.

All existing coordinate systems, with a few notable exceptions [9, 8], have been evaluated only on data sets chosen, constructed, and filtered by the authors. Because they

introduce severe inaccuracies [31, 10, 13, 28], nodes with bad connectivity or part of triangle inequality violations were often removed. To limit oscillations in coordinates [9], measured latencies were smoothed by averaging measurements gathered over large periods of time. However, in a wide area environment, such measurement manipulation is not always possible or desirable. Although network coordinates predict distances accurately, it is yet unclear whether the accuracy is the result of design properties of the coordinate system or is more related to attributes of the data sets used for testing.

In this paper, we revisit network coordinate approaches with no vested interest in showing them to be effective. Instead, we aim to offer an analysis on how data set properties and characteristics of the embedding space interact in producing good embeddings. To understand the influence of the data sets, we do not necessarily remove nodes that participate in triangle inequality violations or with bad connectivity or smoothen latency samples. To examine the effect of space selection—Euclidean or Hyperbolic—decision, we develop a hybrid approach of the Vivaldi algorithm [4] that uses both Euclidean and Hyperbolic space.

We evaluate the hybrid Vivaldi algorithm on seven sets of latency measurements, three from King [7] measurements between DNS servers and the other four between PlanetLab nodes. Our results reveal that the accuracy of the embedding depends on the latency distribution of each data set and is positively influenced by both node filtering and latency filtering. Furthermore, contrary to what we expected, we find that the performance of the two versions of Vivaldi varies with each data set. Hyperbolic coordinates underestimate large latencies (> 100 ms) but are more accurate and comparable to Euclidean coordinates in estimating distances between closer nodes.

Our contributions can be summarized as follows:

- We show relationships between the properties of a coordinate system and the inherent characteristics of the data sets on which it is used. Our results reveal that *existing systems work well on the data sets that were tuned for them, but not so well with third party data sets that exhibit different characteristics.*
- We propose two distributed embedding heuristics that use both Euclidean and Hyperbolic coordinates and achieve good accuracy for **all** data sets. Our results offer *insight into building coordinate systems that*

are both **robust and accurate** in Internet-like environments.

- We construct a hybrid Vivaldi algorithm that embeds nodes simultaneously in both Hyperbolic and Euclidean spaces. In doing so, we are *the first to compare Euclidean and Hyperbolic embeddings in a distributed setting and using a data set with more than 200 nodes.*

The rest of the paper is organized as follows. Section 2 reviews design decisions that impact the accuracy of existing coordinate systems. In Section 3 we describe the Vivaldi algorithm and the data sets used for evaluation. We present the effects of node and latency filtering on the accuracy of Euclidean Vivaldi in Section 4. In Section 5, we compare the accuracy of the embeddings using both Euclidean and Hyperbolic coordinates. Based on observations from Sections 4 and 5 we propose distributed heuristics that leverage the best of both spaces in Section 6. We conclude in Section 7.

2 Related Work

In this section, we review design decisions that impact the accuracy of network embedding. Existing coordinate systems have three important components to their designs: *space selection*, *probing* and *positioning*. Space selection involves how to calculate the distances between points, whether the space is Euclidean, how many dimensions it has, etc. Probing is the process of measuring latency to a few peers, or chosen landmarks. Positioning is the optimization process of using probe results to assign a location to every node in the space. We focus here on space selection and probing.

The latencies between pairs of nodes in the Internet do not form a metric space, mainly because of asymmetric routing and triangle inequality violations. Embedding Internet nodes into a metric space negatively impacts the accuracy of the embedding. Several research efforts have tried to determine the space that best fits the Internet. GNP [15], Lighthouses [17], Virtual Landmarks [26], ICS [11], PIC [3] and Vivaldi [4] use an n-dimensional Euclidean coordinate space, motivated by the fact that latencies in the Internet are dominated by geographic distance. Dabek *et al.* [4] propose two other models. Spherical coordinates are motivated by the fact that the modeled distances are computed on the spherical surface of the Earth. However, since paths in the Internet do not wrap around the Earth, the spherical model has been abandoned for a simpler Euclidean space. The height model augments 2-dimensional Euclidean spaces with a height that captures the time needed to traverse the access links from a node to the core of the Internet. Lee *et al.* [10] add a localized adjustment term to Euclidean coordinates to account for the non-Euclidean effect of triangle inequality violations. Shavitt and Tenkel [21] propose Hyperbolic spaces, motivated by the jellyfish structure—a core in the middle with many tendrils—of the Internet [27]. We show how Euclidean and Hyperbolic embeddings compare to each other [12] but also how they interact with measurement manipulation techniques like node and latency filtering.

The measurement data collected is very important in allowing nodes to position themselves correctly in the embedding space. Ledlie *et al.* [9, 8] find that unfiltered latencies disrupt the Vivaldi algorithm and by removing some latency samples, the coordinates become more stable and accurate. Nodes with bad connectivity or part of severe triangle inequality violations have generally been removed from measurements [4, 28]. However, it is unclear what system designers should do when measurement manipulation is not possible or not desired. We propose heuristics that require no *a priori* filtering and still perform well; accuracy is still good without artificially removing nodes and links from the calculations.

3 Methodology

3.1 Vivaldi

We chose Vivaldi as our network coordinate system because it is distributed and adaptive, running without global state and accommodating the dynamics of the network. Vivaldi simulates a system of springs where each spring corresponds to a pair of nodes. The rest length of a spring emulates the real distance between two nodes while the actual length is the distance computed by the embedding. The energy of each spring is proportional to its displacement (the difference between the rest length and the current length). The algorithm runs iteratively at each node and simulates the progress of the springs toward a state with minimum energy. At every step, each node will be pushed to a new position that minimizes the displacement of the springs it is connected to. Two factors affect the position of a node after each step: the magnitude (M) and the direction (D) of movement. M is proportional to the displacement of the associated springs and D is the opposite of the gradient of the energy function with respect to the position of the node. After computing the magnitude and the direction of movement the following rule updates the coordinates of a node:

$$x = x + \delta \times M \times D$$

where δ is the timestep between two consecutive updates.

We consider an n-dimensional coordinate space. The energy of a spring between nodes x and y is:

$$E_{xy} = \frac{1}{2}k(rtt_{xy} - d(x,y))^2$$

where $rtt_{xy} - d(x,y)$ is the displacement of the spring¹ and k is the elasticity constant.

Dabek *et al.* [4] use coordinates in Euclidean space augmented with a height value h . The distance between two nodes x and y , with $x = (x_1, x_2, \dots, x_n, h_x)$ and $y = (y_1, y_2, \dots, y_n, h_y)$, is:

$$d(x,y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2 + h_x + h_y}$$

After computing the gradient of the energy function, they obtain the following expressions for M and D :

¹We use subscripts to differentiate between a measured value (rtt_{xy}) and a computed value ($d(x,y)$)

Name	Source	Nodes	RTT (ms)		Hyp. Curv.	TIVs	Duration	When	Filtering	
			Avg.	Med.					Node	Latency
PL-Vivaldi	[24, 4]	192	193	152	-20	5%	1 mo.	2004	✓	✓
PL-Sidecar	[22]	384	250	194	-40	14%	1 hr.	5/2006		
King-Meridian	[7, 29]	2500	79	56	-11	8%	1 wk.	5/2004	✓	✓
King-Vivaldi	[7, 4]	1953	326	284	-32	5%	1 wk.	6/2006		✓
PL-Vivaldi-Lat		139	92	90	-16	4%	1 mo.	2004	✓	✓
PL-Sidecar-Filt		262	205	172	-34	13%	1 hr.	5/2006	✓	
King-Vivaldi-Filt		1740	181	158	-16	4%	1 wk.	6/2006	✓	✓

Table 1. Data Sets. Columns represent: (1) the data set name, (2) the publication that describes the measurement and filtering methodology, (3) the number of nodes, (4) the average RTT, (5) the median RTT, (6) the hyperbolic curvature that produces the most accurate embedding (Section 5.3), (7) the percentage of triples that violate the triangle inequality, (8) the duration of data collection, (9) when the data was gathered, (10) whether nodes were filtered (Sections 4.2 and 5.6) and (11) whether latency samples were filtered (Sections 4.2 and 5.6).

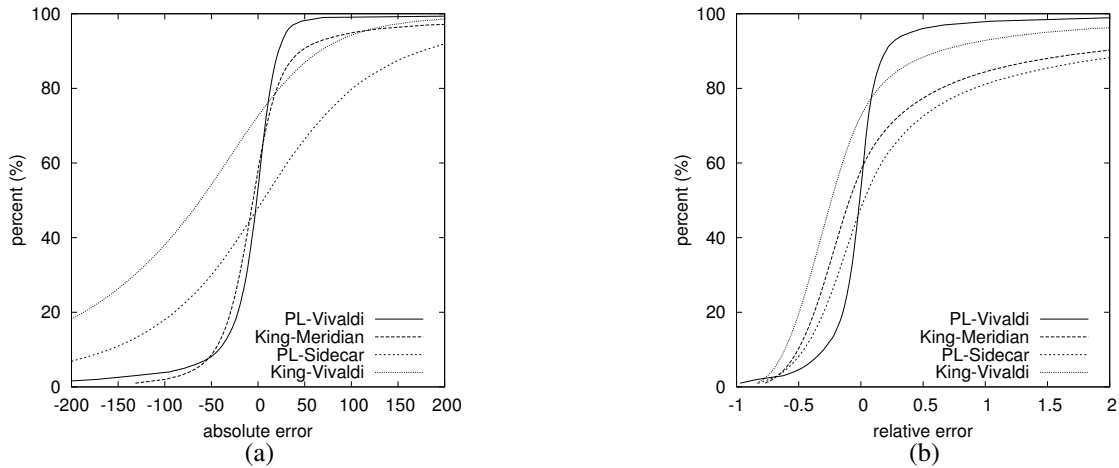


Figure 1. Cumulative distributions of (a) absolute and (b) relative errors for the four main data-sets for Euclidean embedding

$$M = \frac{rtt_{xy} - d(x, y)}{\sqrt{\sum_{i=1}^n (x_i - y_i)^2}} \quad D = u(x - y)$$

where u is a unit vector.

3.2 Data Sets and Experiment Setup

We use seven data sets: four main data sets and three derived from the main sets, summarized in Table 1. King data sets consist of latencies between DNS servers; PL data sets between PlanetLab nodes. We use this diversity of data sets to capture the behavior of the embedding in different environments. Details of the measurements can be found in the references in the table.

We use the MIT p2psim [16] packet-level network simulator. Each node has 32 neighbors: half selected as the closest peers in network latency and the rest chosen at random. Each node starts at the origin of the space, and moves using Vivaldi’s adaptive timestep to converge quickly. After ten seconds, we stop the algorithm and collect the coordinates; longer runs did not significantly improve accuracy.

4 The Importance of Measurement Manipulation

In this section we analyze the accuracy of the embeddings using error distributions and show how characteristics of the data sets can influence the final results. We position nodes in an Euclidean space because distances in the Internet are dominated by geographic distance and paths do not generally “wrap around” the Earth. In Section 5 we also experiment with Hyperbolic spaces. We choose a two-dimensional Euclidean space augmented with heights due to its simplicity and because it was shown to produce good embeddings [4]. We experimented with higher-dimension spaces and obtained similar results.

4.1 Error Distributions

We evaluate the accuracy of Euclidean Vivaldi by absolute and relative errors computed over all pairs of nodes in the four main data sets. Absolute error is the difference between the embedded distance and the real distance; relative error is the absolute error divided by the real distance. We do not use absolute values for these two measures (as in other work) because we want differentiate between under-

and over-estimation. Other accuracy metrics, like relative rank loss [12] better capture the usefulness of the embedding for applications, but by relying only on the relative distance to nodes they tend to overstate the importance of small errors.

Figure 1(a) presents the distribution of the absolute error for the four main data sets. Each point corresponds to one pair of nodes. Euclidean Vivaldi exhibits the best performance on the PL-Vivaldi data set, with more than 90% of the pairs having an error within the range $[-50,50]$. For the PL-Sidecar and King-Vivaldi data sets, contrary to what we expect, the embedding performs much worse. Only 60% of the pairs have an absolute error within the range $[-100,100]$, compared to more than 80% in the first two data sets.

We also plot the distribution of relative errors in Figure 1(b). A relative error of 1 between a pair of nodes means that the embedded distance is twice the real distance; of -0.5 that the embedded distance is half the real distance. We summarize the error prediction results of the two embeddings in Table 2 (ignore the columns labeled “Hyp” for now).

Of the four main data sets, Vivaldi performs worst on PL-Sidecar and King-Vivaldi. For example, in PL-Sidecar, only 39% of the distances are predicted to be within 25% of the real distances (columns 2 and 3 of Table 2). This PL-Sidecar data set was created by averaging RTTs between PlanetLab nodes over a period of an hour, as opposed to the other main data sets, built from measurements gathered over the course of at least a week. Further, unlike King-Meridian and PL-Vivaldi, no nodes were removed due to bad measurements or connectivity problems. On the other hand, King-Vivaldi has the worst underestimation error: close to 20% of all distances are estimated to be less than half of the real distances, compared to around 10% for the other three data sets. More, 72% of all distances are underestimated in King-Vivaldi, unlike the other main data sets where at most 58% of the predictions are less than the real latency. After contacting the creators of the King-Vivaldi set and carefully examining the latencies, we found that it contains several nodes that appear to be very close to all other nodes. They create bad triangle inequality violations and may be the source of the underestimation.

From these results, we hypothesize that *both latency and node filtering improve the quality of the embedding*. While latency filtering has already been shown to improve embedding accuracy [9], we seek to understand the influence of node filtering next.

4.2 Node Filtering

Latency and node filtering can improve the embeddings. Ledlie *et al.* found that unfiltered latencies disrupt the Euclidean Vivaldi algorithm [9]. By removing some latency samples, the solution becomes more stable and accurate. Wang *et al.* obtain better performance with Euclidean Vivaldi when they do not probe edges that are part of severe triangle inequality violations [28]. Yet, common practice is to also filter out **nodes** that participate in many triangle inequality violations. Here we focus on node filtering and ask how the removal of certain nodes affects the accuracy of the

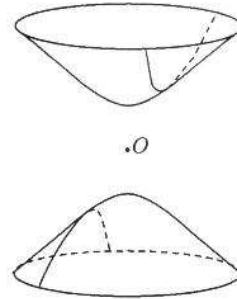


Figure 2. Hyperboloid with two sheets: points in the Hyperbolic space use only the upper sheet, O is the origin.

embedding.

We use two data sets. Dabek *et al.* generated King-Vivaldi-Filt from the King-Vivaldi data set by removing the nodes with severe triangle inequality violations. We apply the same approach, removing 132 nodes from PL-Sidecar to produce PL-Sidecar-Filt. Table 1 includes these data sets; Table 2 presents the resulting accuracy statistics.

As expected, Euclidean Vivaldi performs much better on King-Vivaldi-Filt than on King-Vivaldi. After removing the outliers, more than 70% of the distances are predicted to be within 25% of the real values, an increase of almost double. Outliers in the node set can disrupt the embedding considerably. For the PL-Sidecar-Filt data set, however, the improvement is smaller, indicating that latency filtering, which was not applied to this data set, plays a significant role in the embedding process.

5 The Importance of Space Selection

In the previous section we have shown that the accuracy of the coordinates depends significantly on the filters applied to them. Next we seek to understand how the characteristics of the embedding space impact the behavior of the embedding and how can we use the results to obtain more robust coordinate systems. We compare the accuracy results obtained when embedding nodes in Euclidean and Hyperbolic spaces. Hyperbolic coordinates are motivated by several studies showing that the Internet has a jellyfish structure (a core in the middle and many tendrils connected to it) [27, 25]. In a Hyperbolic space the distance between two points is computed along a curved line bent towards the origin. The closer to the origin the points are, the shorter the distance between them is. Similarly, in the Internet, the farther two nodes are from the core, the longer the path between them is. Shavitt and Tankel [21] and Lua *et al.* [12] have shown good accuracy results when embedding nodes into a Hyperbolic space.

5.1 Hyperbolic Spaces

It is hard for an observer in the three-dimensional Euclidean world to visualize the Hyperbolic world [1]. To better understand it, different Euclidean models of Hyperbolic space have been constructed. Unfortunately, each model of

Data Set	% of all predicted dist						% of underestimated dist				% of overestimated dist			
	within 25% of real		within 50% of real		within 100% of real		of all		< 1/2 real		of all		> 2 x real	
	Euc	Hyp	Euc	Hyp	Euc	Hyp	Euc	Hyp	Euc	Hyp	Euc	Hyp	Euc	Hyp
PL-Vivaldi	76	44	90	73	97	90	58	50	5	7	42	50	3	19
PL-Vivaldi-Lat	77	72	90	86	97	97	47	56	4	5	53	44	3	3
PL-Sidecar	39	32	63	60	81	76	48	44	8	7	52	56	19	23
PL-Sidecar-Filt	49	35	70	62	83	76	50	48	5	6	50	52	17	24
King-Vivaldi	34	33	68	70	92	92	72	72	19	16	28	28	8	7
King-Vivaldi-Filt	71	40	90	79	98	96	55	73	5	12	45	27	2	4
King-Meridian	38	36	67	69	84	89	57	63	9	13	43	37	16	10

Table 2. Summary of prediction errors for the seven data sets. We show, for each Embedding and each data set, (1) the percentage of distances estimated to be within 25%, 50% and 100% of the real distances (higher values are better), (2) the percentage of all underestimated distances and of the distances predicted to be less than half the real distances (lower values are better), (3) the percentage of all overestimated distances and of the ones estimated as more than twice the real distances (lower values are better).

Hyperbolic geometry is distorted, just as a two-dimensional map is a distorted depiction of the Earth.

To describe the Hyperbolic space, all we need to know is the amount of distortion introduced by the model used to embed the space. The distortion is determined by two parameters: curvature and metric. *Curvature* is a characteristic of the space and represents the amount by which an object in the space deviates from being flat: Euclidean spaces have curvature 0 because all lines are flat; Hyperbolic spaces have negative curvature. The *metric* is a characteristic of the model used to embed the space and represents the distance function between two points. Knowing the curvature of a Hyperbolic space and the metric of its model we can determine the distance between any two points in the space as well as the curved line along which the distance is computed.

Of several equivalent models of the Hyperbolic world, we choose the hyperboloid, in which all points lie on the upper sheet of a hyperboloid, due to the simplicity of its metric. Figure 2 presents a hyperboloid with two sheets. Although the model stretches to infinity, it is cut for observation. The distance between two points on the hyperboloid is computed along a line formed by the intersection of the hyperboloid with the plane determined by the two points and the origin.

5.2 Hyperbolic Vivaldi

We now consider the original Vivaldi protocol where the nodes and the conceptual springs lie instead in a Hyperbolic space. The spring placed between two points will move along a curved line under a force proportional to the difference between the real distance and the Hyperbolic distance.

The distance between two points $x = (x_1, x_2, \dots, x_n)$ and $y = (y_1, y_2, \dots, y_n)$ in a n-dimensional Hyperbolic space of curvature k is:

$$d(x, y) = \operatorname{arccosh} \left(\sqrt{\left(1 + \sum_{i=1}^n x_i^2\right) \left(1 + \sum_{i=1}^n y_i^2\right) - \sum_{i=1}^n x_i y_i} \right) \times |k|$$

This formula is derived from the distance between two points on the unit hyperboloid (the *arccosh* part) multiplied

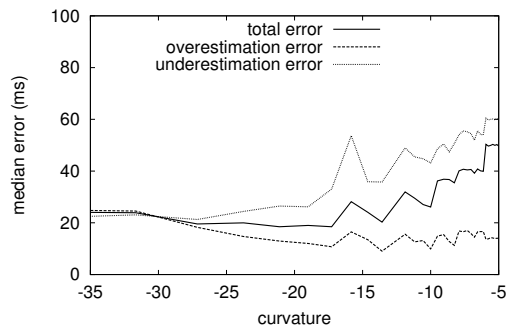


Figure 3. Median error of the embedding versus space curvature.

by the absolute value of the curvature of the Hyperbolic space (k). The curvature factor is similar to the unstretch value used by Shavitt and Tankel in Hyperbolic BBS [21] to normalize the distance on the hyperboloid.

The gradient of the energy function provides the magnitude M and the direction D that guide the movement of a node at each iteration of the algorithm, similarly to Euclidean Vivaldi (Section 3).

$$D = u \left(\frac{\sqrt{1 + \sum_{i=1}^n y_i^2} x - y}{\sqrt{1 + \sum_{i=1}^n x_i^2}} \right)$$

$$M = \frac{rtt_{xy} - d(x, y)}{\sqrt{\cosh^2 d(x, y) - 1}}$$

We evaluate the performance of Vivaldi in Hyperbolic space using the data sets in Table 1. We modified the Vivaldi code to maintain both Euclidean and Hyperbolic coordinates concurrently but left the rest of the protocol and its parameters unchanged (see Section 3.2). The Hyperbolic coordinates have three dimensions while the Euclidean space remains two-dimensional with heights. Thus, a node has the same number of degrees of freedom in both spaces.

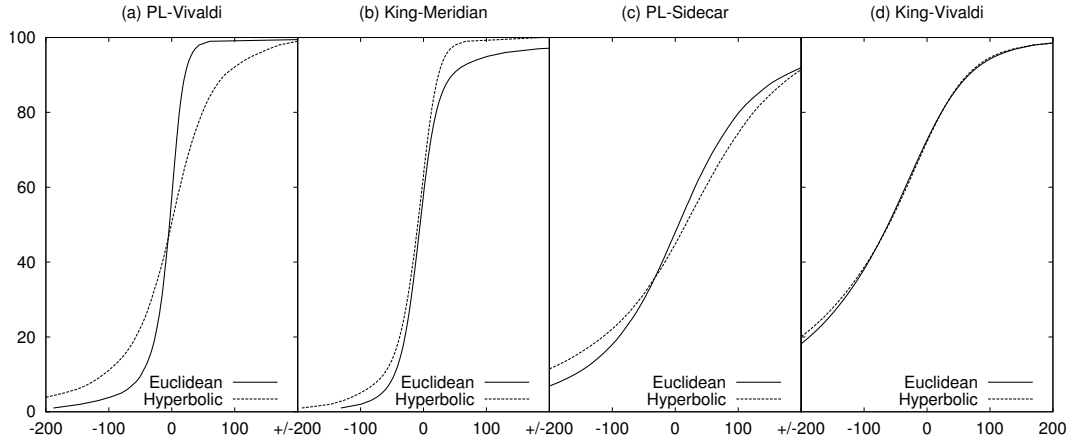


Figure 4. Cumulative distribution of absolute errors (in milliseconds) for the main data-sets for different embeddings.

5.3 Curvature of the Space

To provide the best chance for Hyperbolic embeddings, we choose the best-fit curvature value—the curvature that minimizes median error—which depends on the embedded latencies and varies by data set. Figure 3 shows how curvature affects the accuracy of the embedding on the PL-Vivaldi data set. The horizontal axis is the curvature value and the vertical axis the median absolute error of the embedding (the difference between the embedded and real distances). Figure 3 shows median error decreases to a plateau at approximately -17 and is relatively stable until -27.

To better analyze this relationship, we separate overestimation error—the median error when embedding distance is greater than the real distance—from underestimation error—the median error of the rest. Underestimation error decreases as curvature becomes increasingly negative. Overestimation error decreases from curvature -32 to -17, then oscillates between 10 and 15 ms. Any curvature between -27 to -17 will produce a low embedding error. We choose the curvature for the embedding of the PL-Vivaldi data set to be -20. Based on similar results, we set the curvatures for every other data set independently because we want to obtain the best possible Hyperbolic embeddings.

5.4 Error Distributions

Similarly to Vivaldi in Euclidean spaces, we use absolute and relative errors to evaluate the accuracy of Vivaldi in Hyperbolic spaces. We focus first on the four main data sets and present the absolute error distributions in Figure 4. We replot the error distributions of Euclidean Vivaldi to better compare the two approaches. For the PL-Vivaldi data set, both embeddings have as many over-estimated distances as under-estimated. However, Hyperbolic Vivaldi has higher error in both. 10% of the pairs are estimated to be over 100 ms closer than the real distance when embedded into Hyperbolic space, compared to fewer than 5% in Euclidean space. On the other hand, the accuracy of Hyperbolic Vivaldi improves visibly when simulated on the King-Meridian data set. For the PL-Sidecar and King-Vivaldi data sets, similarly to Euclidean Vivaldi, the performance of Hyperbolic

Vivaldi is worse than for the first two data sets. However, only the PL-Sidecar data set has as many under- as over-estimated distances. For the King-Vivaldi latencies the ratio is skewed, with 65% of pairs underestimated compared to only 35% overestimated.

Figure 5 presents the distributions of relative errors. The Hyperbolic embedding is less accurate than the Euclidean embedding on all main data sets except King-Meridian—the one that has the smallest median RTT. Table 2 confirms the observation: 16% of the distances in the Euclidean embedding are estimated to be at least twice as much as the real values, compared to only 10% in the Hyperbolic embedding. Furthermore, Figures 4 and 5 also show that there is almost no difference between the relative underestimation error of the two algorithms over all data sets, as opposed to the absolute underestimation error. This discrepancy could be because Hyperbolic Vivaldi underestimates *more* distances than Euclidean Vivaldi but most of these distances are *large*, causing large absolute errors but not large relative errors. On the other hand, Euclidean Vivaldi underestimates smaller distances and although the absolute error is smaller, it is significant. That Hyperbolic Vivaldi performs best on the data set that has shorter distances suggests that it is better-suited to low-latency data sets and to predicting smaller latencies.

From these results, we hypothesize that *Hyperbolic Vivaldi performs much better for shorter distances than longer distances*. We evaluate this hypothesis next.

5.5 Latency Distribution

To verify that Hyperbolic Vivaldi performs better on shorter distances, we construct a new data set with lower latencies between nodes. We choose PL-Vivaldi and remove all nodes that have more than 25% of the RTTs to each of the other nodes higher than 100 ms. This produces the PL-Vivaldi-Lat data set, composed of 139 nodes, with median and mean RTT of 90 and 92 ms. These values are much closer to those that characterize the King-Meridian data set. We chose PL-Vivaldi as the initial data set because it exhibits the greatest difference in accuracy between the Eu-

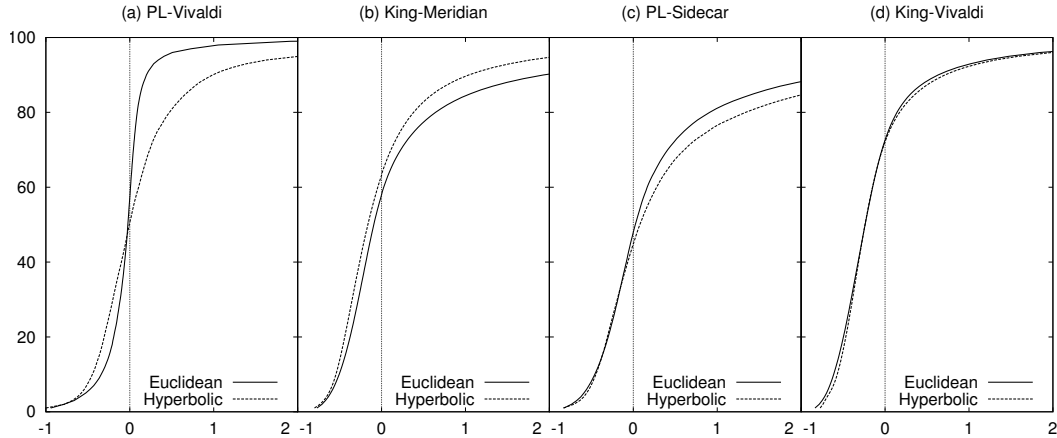


Figure 5. Cumulative distribution of relative errors for the main data-sets for different embeddings.

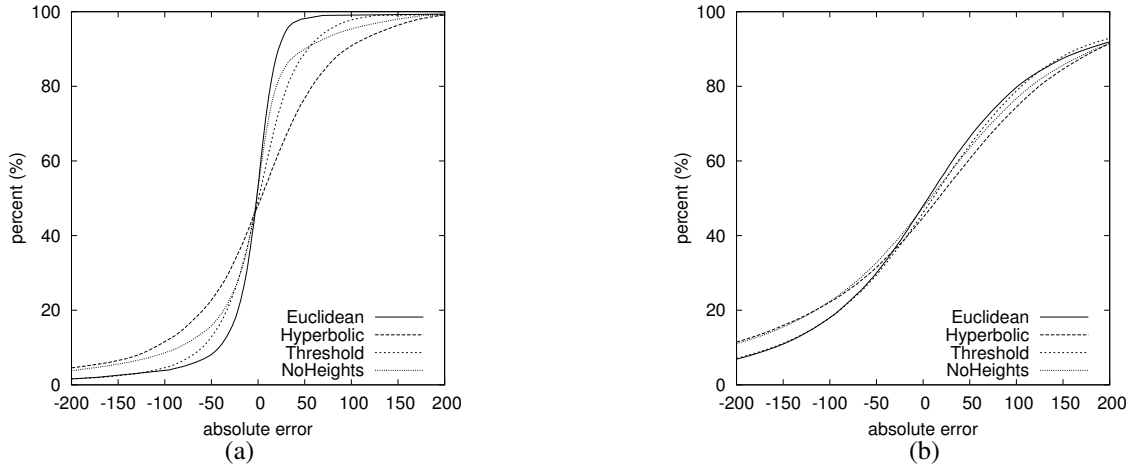


Figure 6. Cumulative distributions of absolute errors for (a) PL-Vivaldi and (b) PL-Sidecar data-sets for different embeddings

clidean and Hyperbolic embeddings. We compute the error distributions for PL-Vivaldi-Lat and summarize the results in Table 2. As expected, while Euclidean Vivaldi performs similarly for the new data set, the accuracy of the Hyperbolic embedding improves considerably: 86% of the pairs of nodes are predicted to have an RTT within 50% of the real RTT, compared to only 73% for PL-Vivaldi.

That Hyperbolic embeddings tend to underestimate large latencies, but perform much better for shorter distances can be used to construct new and efficient hybrid embeddings. These embeddings would benefit from both Euclidean and Hyperbolic coordinates and would achieve *good* accuracy for *all* data sets, as described in Section 6. They could be particularly useful for applications, such as overlay multicast, server selection or overlay construction, that have been shown to suffer from the high embedding errors on short links [30].

5.6 Node Filtering Revisited

To evaluate how node filtering affects the Hyperbolic embedding, we use King-Vivaldi-Filt and PL-Sidecar-Filt.

Table 2 shows that the accuracy of the Hyperbolic embedding does not improve as much as that of Euclidean when nodes are filtered in either of the data sets. This suggests that Hyperbolic coordinates are less sensitive to node filtering and thus may be good candidates where node filtering is not desired or possible.

6 Embedding Heuristics

We want to obtain an embedding algorithm that benefits from both Euclidean and Hyperbolic coordinates. Ideally, when it estimates the latencies to other nodes, a node in our system would select to use the coordinates and distance function that yield the smallest error. Unfortunately, this approach is unfeasible in a distributed setting where nodes make decisions without global knowledge and where the number of measurements is limited. To address this, we propose two heuristics that allow a node to choose the best coordinates based only on local information.

In the first heuristic, called *NoHeightsHyperbolic*, the estimated distance between each pair of nodes is computed using the Euclidean metric whenever both nodes in the pair

have the height at most 1. When at least one of the nodes has a height greater than 1, we use the Hyperbolic distance. The intuition behind this heuristic is drawn from the initial motivation for height vectors by Dabek *et al.* [4]. The Euclidean distance models the Internet core where latencies are proportional to geographic distances, while the height accounts for the time taken to reach the core from a node behind an access link. By using Hyperbolic distance estimation whenever a node has height, we offer an alternative to the Vivaldi's height model.

The second heuristic, *ThresholdHyperbolic*, is partly based on the observations made in Section 5. Since Hyperbolic space seems to underestimate larger latencies, we choose Euclidean distance estimation for every pair whose Hyperbolic distance is computed to be more than a certain threshold. In the experiments we used a threshold of 100ms.

We show results for the PL-Vivaldi and PL-Sidecar data sets in Figure 6. The absolute error distributions of the embeddings using the two heuristics are labeled *NoHeights* and *Threshold*. The *ThresholdHyperbolic* embedding improves the accuracy of the pure Hyperbolic embedding and, as expected, eliminates almost completely the difference in underestimation when compared to the Euclidean embedding. Although it does not perform better than Euclidean Vivaldi, it proves to be a more general alternative due to its good accuracy for all data sets. Due to the Hyperbolic component, it is also more robust than a pure Euclidean embedding when node filtering is not used.

7 Conclusions

We have shown how the inherent properties of data sets used in network coordinates papers can interact with the space selection process to produce better embeddings. We adapted the Vivaldi algorithm so that nodes position themselves in a Hyperbolic space and we compared the performance of Euclidean and Hyperbolic embeddings using seven different data sets. Our results show that the accuracy of the two versions of Vivaldi varies with each data set and that Hyperbolic coordinates have the tendency to underestimate large latencies. Further, the accuracy of the results depends significantly on the latency distribution of the data sets and on the filters applied to them. Based on these results we presented a distributed heuristic, *ThresholdHyperbolic*, that uses both Euclidean and Hyperbolic coordinates and achieves good accuracy for **all** data sets. We believe that our observations provide insight into how to build more robust and more accurate distributed coordinate systems.

References

[1] E. A. Abbott. *Flatland: A Romance of Many Dimensions*. publisher unknown, 1880.
 [2] B. Cohen. Incentives build robustness in BitTorrent. In *P2PEcon*, 2003.
 [3] M. Costa, M. Castro, A. Rowstron, and P. Key. PIC: Practical Internet coordinates for distance estimation. In *ICDCS*, 2004.
 [4] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM*, 2004.

[5] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with Coral. In *NSDI*, 2004.
 [6] Gnutella. <http://www.gnutella.com>.
 [7] K. Gummadi, S. Saroiu, and S. Gribble. King: Estimating latency between arbitrary Internet end hosts. In *IMW*, 2002.
 [8] J. Ledlie, P. Gardner, and M. Seltzer. Network coordinates in the wild. In *NSDI*, 2007.
 [9] J. Ledlie, P. Pietzuch, and M. Seltzer. Stable and accurate network coordinates. In *IEEE ICDCS*, 2006.
 [10] S. Lee, Z.-L. Zhang, S. Sahu, and D. Saha. On suitability of euclidean embedding of internet hosts. In *Sigmetrics*, 2006.
 [11] H. Lim, J. C. Hou, and C.-H. Choi. Constructing internet coordinate system based on delay measurement. In *IMC*, 2003.
 [12] E. K. Lua, T. Griffin, M. Pias, H. Zheng, and J. Crowcroft. On the accuracy of the embeddings for Internet coordinate systems. In *IMC*, 2005.
 [13] C. Lumezanu, D. Levin, and N. Spring. PeerWise discovery and negotiation of shorter paths. In *HotNets*, 2007.
 [14] Y. Mao and L. K. Saul. Modeling distances in large-scale networks by matrix factorization. In *IMC*, 2004.
 [15] T. S. E. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *INFOCOM*, 2002.
 [16] P2PSim. <http://pdos.csail.mit.edu/p2psim>.
 [17] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti. Lighthouses for scalable distributed location. In *IPTPS*, 2003.
 [18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM*, 2001.
 [19] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of IFIP/ACM Middleware*, 2001.
 [20] Y. Shavitt and T. Tankel. Big-bang simulation for embedding network distances in euclidean space. In *INFOCOM*, 2003.
 [21] Y. Shavitt and T. Tankel. On the curvature of the Internet and its usage for overlay construction and distance estimation. In *INFOCOM*, 2004.
 [22] R. Sherwood and N. Spring. Touring the Internet in a TCP sidecar. In *IMC*, 2006.
 [23] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *SIGCOMM*, 2001.
 [24] J. Stribling. Planetlab all pairs ping. http://www.pdos.lcs.mit.edu/strib/pl_app/.
 [25] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the internet hierarchy from multiple vantage points. In *Proceedings of INFOCOM*, 2002.
 [26] L. Tang and M. Crovella. Virtual landmarks for the internet. In *IMC*, 2003.
 [27] S. Tauro, C. Palmer, G. Siganos, and M. Faloutsos. A simple conceptual model for the internet topology. In *Global Internet*, 2001.
 [28] G. Wang, B. Zhang, and T. S. E. Ng. Towards network triangle inequality violation aware distributed systems. In *IMC*, 2007.
 [29] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A lightweight network location service without virtual coordinates. In *ACM SIGCOMM*, 2005.
 [30] R. Zhang, C. Tang, Y. C. Hu, S. Fahmy, and X. Lin. Impact of the inaccuracy of distance prediction algorithms on Internet applications—an analytical and comparative study. In *IEEE Infocom*, 2006.
 [31] H. Zheng, E. K. Lua, M. Pias, and T. G. Griffin. Internet routing policies and round-trip times. In *PAM*, 2005.