# ANEDA: Adaptable Node Embeddings for Shortest Path Distance Approximation

Frank Pacini
*Department of Computer Science*
*Boston University*
Boston, USA
fgpacini@bu.edu

Allison Gunby-Mann
*Thayer School of Engineering*
*Darmouth College*
Hanover, USA
allison.mann.th@dartmouth.edu

Sarel Cohen
*Hasso Plattner Institute*
Potsdam, Germany
sarel.cohen@hpi.de

Peter Chin
*Thayer School of Engineering*
*Darmouth College*
Hanover, USA
pc@dartmouth.edu

***Abstract*—Shortest path distance approximation is a crucial aspect of many graph algorithms, in particular the heuristic-based routing algorithms that make fast, scalable map navigation possible. Past literature has introduced deep learning models which try to approximate these distances by training on graph embeddings (i.e. node2vec, Gra100, ProNE, Poincare). We propose ANEDA, a more lightweight technique than the embedding and graph neural network scheme, which involves training the embeddings directly, using either previous embedding techniques or geographic coordinates as a good initialization. We demonstrate the applications ANEDA to deep A* routing and learned road maps. Through experiments on several road and social networks, we show our model's error reduction of up to 75% against two recent deep learning approaches, and its competitive performance against the larger, state-of-the-art architecture.**

***Index Terms*—graph embedding, shortest path distance, neural networks**

## I. Introduction

Given a graph $G = (V, E)$ and an embedding dimension $D$, the goal of this line of research is to find a combination of an embedding map $\phi : V \to \mathbb{R}^D$ and a distance measure $M : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}_+$ such that for nodes $u, v \in V$ and distance between them $d_{u,v}$,

$$\hat{d}_{u,v} = M\big(\phi(u), \phi(v)\big) \approx d_{u,v}.$$

Finding such an embedding map is useful because we can represent the graph by effectively approximating the $|V| \times |V|$ adjacency matrix, which may be infeasible to store or calculate for large graphs, in a $|V| \times D$ embedding matrix containing the vectors $\phi(v)$. Querying for a distance approximation would then take $O(1)$ to retrieve $\phi(v)$ and (for reasonable choices) $O(D)$ to apply $M$.

### A. Heuristic-based Routing

An important application for embedding matrices is network routing. Dijkstra's is the well-known standard algorithm for finding shortest path routes between a source node $u$ and target $v$, however, in practice, it is too slow to be feasible for large-scale routing or very large networks. A$^*$ [1] is an optimization to Dijkstra's that uses an adjustable heuristic function $h$ to improve performance, at the (usually small) sacrifice of finding an approximate shortest path. In order to prioritize visiting nodes that are expected to bring the algorithm closer to the target $v$, A$^*$ queries $h$ for approximations of the distance between each candidate node $w$ and $v$. A$^*$'s performance is heavily dependent on the quality of $h$, so we want to provide $A^*$ with a distance approximation function with as minimal prediction error as possible, given the distance data that we can feasible calculate for $G$. Figure 1 illustrates the impact of heuristic choices by comparing nodes visited in a route navigation task between Dijkstra's (which is equivalent to A$^*$ with $h = 0$), $A^*$ with $h =$ the geographic distance between the nodes and $A^*$ with an embedding model from early in our investigation.

### B. Node Embedding Techniques

Common approaches to produce an embedding map $\phi$ for graphs include various random walk strategies [2], [3], [4], [5], [6], [7] which define node representations based on co-occurrence in these walks, and matrix factorization techniques [8], [9], [10], [11] which manipulate the graph adjacency matrix to more explicitly represent multi-level graph structure. We focus our investigation on two well-established techniques utilizing each approach: Node2Vec [3] and GraRep [9]. Node2Vec utilizes the Skip-Gram model to generate embeddings, which encodes pair associations as a function of the dot product. GraRep on the other hand computes and then merges $k$-step transition probability matrices of the node pairs to produce their final representations.

Figure 2 from [12] visualizes the difference between pairs of embeddings generated by Node2Vec and GraRep on the Winterthur, Switzerland OSMnx [13] graph, where the shortest path distances have been divided by the graph diameter. In GraRep (with order 100), we can see that the embedding difference approximates the distribution of graph distances fairly well, whereas Node2Vec produces larger embedding distances that may result in faster training when used in our model.

### C. Distance Approximation Networks

An important issue when trying to use node embeddings for distance approximation is that most well-known techniques are not explicitly designed for distance preservation, and are instead evaluated on more general downstream tasks such as classification or clustering. In response, several deep learning

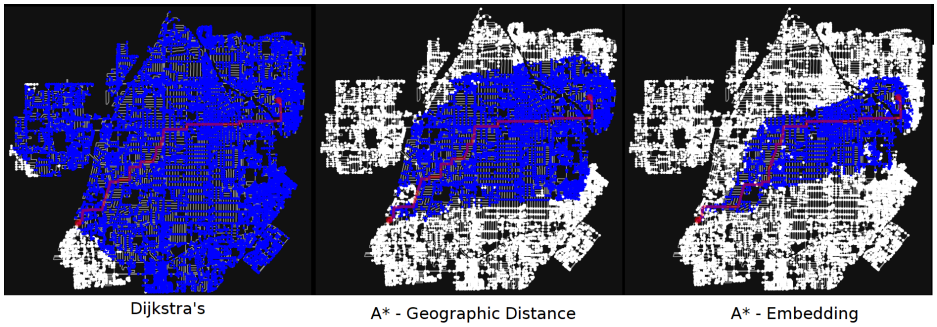


Fig. 1. Santa Ana, CA routing visits with different A* heuristics

architectures have been proposed to train on graph embedding inputs as a strategy for distance approximation [14], [15], [12]. They formulate the distance approximation problem as a machine learning task by simply precomputing the shortest path distances on a subset of node pairs, and using the model to generalize to a full representation matrix. These approaches all use an existing graph embedding technique for $\phi$ and then try to find an optimal measure $M$ using deep learning, or train both jointly.

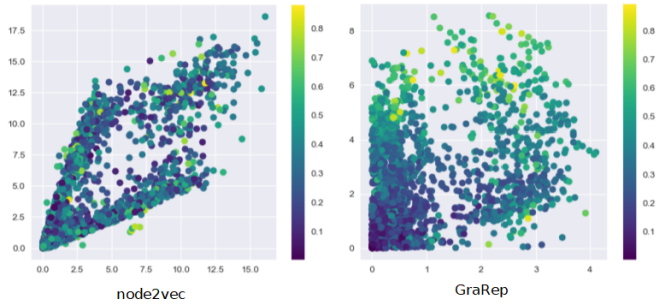


Fig. 2. WTUR 2d vector difference with SP distance labels

## II. ANEDA

As opposed to previous networks, our approach is instead to explicitly choose a function $M$ for the vectors, initialize $\phi = \phi_0$ from another graph embedding and then try to find an optimal $\phi$ through training. Our model is therefore an embedding matrix similar in spirit to Node2Vec, initialized with a graph embedding not explicitly trained for distance approximation. We then tune this initialization in order to get improved shortest path approximations with respect to a chosen distance measure, such as the Euclidean distance or dot product. We focus our research on finding good choices for initialization (**II-B**) and distance measure (**II-C**). For simplicity, in the rest of the paper we also denote $\mathbf{u} = \phi(u)$ when $\phi$ is our model.

Besides comparable performance to state-of-the-art techniques, we believe the merits of our approach are:

1) Adaptability. Our model can be used to find node representations in a particular space by tweaking the distance measure. This could be useful in order to create distance-preserving graph embeddings in spaces with desired properties, or tailor representation for graphs with known or theorized properties. The model can also be easily exported as a 2d array and queried for downstream tasks.

2) Size. Our model is an embedding matrix of size $|V| \times D$, whereas an approach like Vdist2vec [15] contains an embedding layer of the same size as part of its larger model. This smaller size is likely to improve train and evaluation time. Our model can also optionally be initialized with another embedding of size $|V| \times k$, where $k \leq D$, that can be discarded after initialization. Vdist2vec on the other hand requires a one-hot input of size $2|V|$ per example to train.

3) Simplicity. Our model requires only an embedding matrix and trainer (we use node2vec as our base code), plus a simple distance measure in order to implement the model for a specific task.

### A. Data Processing

In order to produce node pairs with distance labels for the model to train with, we follow a similar approach to [14], which is to randomly select $l$ nodes from the graph to use as landmarks, and compute the distance between this landmark and all other nodes using Dijkstra's. Given landmark $u$ and other node $v$, each training example in our dataset is simply $< u, v, d_{u,v} >$. Since we are tuning the embeddings directly, we only need the node indices as inputs. Our embedding is instead introduced when evaluating the loss, similar to Node2Vec [3], whereas the approach in [14] may need to store the full vector input since it is a function of the node embeddings for $u$ and $v$. This greatly reduces the dataset space requirements of our approach, since each example requires storing 3 values instead of $D + 1$ values.

Another notable difference is [14] makes their validation sets include entirely distinct nodes from the test set. We cannot do this for our model since it would prevent embeddings for

the validation nodes from ever being trained, so we instead remove train landmarks from the validation sets to prevent the two sets from having any of the same pairs. As suggested by [12], for weighted graphs we also normalize the edge weights by dividing by the largest across all edges before computing the distance labels.

### B. Initialization

In order to improve training time and potentially performance, we use two embedding types as initializations for the model.

For any graph, we can train graph embeddings not designed specifically for distance approximation, such as Node2Vec, and then tune our model to produce embeddings that approximate graph distances very well. We test specifically with Node2Vec [16] and GraRep [9] embeddings.

If the graph is geographic, we can also use geographic coordinates of each node as an initialization. We evaluate several geographic networks with either latitude-longitude coordinates or UTM coordinates of the nodes included. For latitude-longitude coordinates, where $\varphi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ = latitude and $\lambda \in [-\pi, \pi]$ = longitude, in general, we convert to 3-dimensional Cartesian coordinates using the identity

$$x = \cos(\varphi)\cos(\lambda), \quad y = \cos(\varphi)\sin(\lambda), \quad z = \sin(\varphi)$$

This is done since differences in latitude and longitude reflect distances differently depending on location, and the change also results in coordinates on the same scale as the normalized graph distances. No changes are made to UTM coordinates other than normalization since these have a pre-applied projection into 2-dimensional Euclidean space.

We make use of $k$-dimensional coordinate vectors when available to initialize the first $k$ of our $D$-dimensional embeddings. The remaining $D - k$ dimensions are initialized at random from a normal distribution with a small variance of $1/D$. This allows us to train embeddings of a much larger size than the 2 or 3 dimensional geographic coordinates, or add additional dimensions to initial embeddings for tuning. In general, though, for trainable graph embeddings (like Node2Vec) we use $D = k$.

The different initialization techniques also offer the opportunity to choose distance measures for training with an expected relationship to the initial embedding. For instance, when using the Euclidean distance for training and latitude-longitude initialization (transformed as mentioned above), the initial distance measure is an underapproximation (although very accurate for small sections of the Earth like cities) of the geographic distance between the nodes. This measure is a reasonable baseline approximation for downstream tasks like routing, as we observe in later experiments. In Figure 3, we visualize how these initial coordinates are projected by the model in a graph where the geographic distance does not perform adequately. This is done by applying the transformation from Lat-Long to Cartesian for the initialization, applying the reverse on the output embeddings and then modifying the graph coordinate attributes. The Santa Ana, CA graph from OSMnx has two "islands" which, except for several bridging edges, are separated by highways that extend outside the city boundaries. The model moves these islands up and farther left and right, while also condensing them. Nodes in regions with many short edges are generally condensed, while holes in the graph are created in areas with fewer, longer streets. This model was trained with $D = k = 3$, so this visualization may not fully reflect the representations learned in embeddings with many additional dimensions.

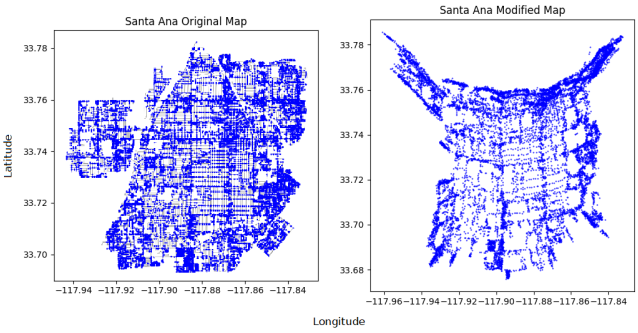


Fig. 3. Initial coordinate modification by ANEDA ($D$ = 3)

### C. Distance Measures

We tested several different distance measures to train the embeddings on, including common norms for Euclidean geometry and measures for hyperbolic and elliptical geometry.

*1) p-norm:* Using several different $p$ values, we tested the $p$-norm of the difference between two node embeddings which is given by

$$\hat{d}_{u,v} = \left( \sum_{i=1}^{D} (\mathbf{u}_i - \mathbf{v}_i)^p \right)^{1/p}.$$

*2) Poincare Hyperbolic:* We additionally use two measures of distance in hyperbolic geometry based on the Poincare Disk and Minkowski Hyperboloid models for hyperbolic space.

In the 2d Poincare Disk model, points are fixed inside the unit disk, with the edge of the disk representing infinite distance. As mentioned in [17], we can extend this idea to the $D$-dimensional unit ball and then the distance measure is

$$\hat{d}_{u,v}(u,v) = \operatorname{arcosh}\left(1 + 2\frac{||\mathbf{u} - \mathbf{v}||^2}{(1 - ||\mathbf{u}||^2)(1 - ||\mathbf{v}||^2)}\right).$$

*3) Minkowski Hyperbolic:* In the Minkowski Hyperboloid model, points are fixed inside the forward sheet of a two sheeted hyperboloid in $D + 1$ dimensional space. All points $\mathbf{x} \in \mathbb{R}^{D+1}$ must satisfy

$$\mathbf{x}_0^2 - \mathbf{x}_1^2 - \cdots - \mathbf{x}_D^2 = 1.$$

Then the distance measure between 2 points $u, v \in \mathbb{R}^{D+1}$ is

$$\hat{d}_{u,v}(u,v) = \operatorname{arccosh}(\mathbf{u}_0\mathbf{v}_0 - \mathbf{u}_1\mathbf{v}_1 - \cdots - \mathbf{u}_D\mathbf{v}_D).$$

In practice, in order to enforce the constraint for each point $\mathbf{x}$, we keep its values in the last $D$ dimensions and set the first dimension, $x_0$ to:

$$x_0 = \sqrt{||\mathbf{x}||^2 + 1}$$

The measure then becomes

$$\hat{d}_{u,v}(u,v) = \cosh^{-1}\left(\sqrt{(||\mathbf{u}||^2+1)(||\mathbf{v}||^2+1)} - \mathbf{u}\cdot\mathbf{v}\right).$$

*4) Elliptic:* We also tested using measures in a positive curvature geometry since this is the true geometry of geographic points. Assuming a radius of 1, the elliptical distance is simply the angle between the points in radians, which can be obtained from the dot product:

$$\hat{d}_{u,v} = \cos^{-1}\left(\frac{\mathbf{u}\cdot\mathbf{v}}{||\mathbf{u}||||\mathbf{v}||}\right).$$

A natural initialization to use with the elliptic distance is geographic coordinates (still with the Cartesian transformation), since this gives the geographic distance exactly, and so should in theory provide a better initialization than with the Euclidean distance measure.

*5) Inverse Dot:* With inspiration from the elliptic distance, we tried several other functions besides inverse cosine which would take the normalized dot product $\delta$ and "invert" it by mapping $\delta = -1$ to a large distance and $\delta = 1$ to 0. We found the most effective to be

$$\hat{d}_{u,v} = \left(1 - \frac{\mathbf{u}\cdot\mathbf{v}}{||\mathbf{u}||||\mathbf{v}||}\right) * d_{max}/2$$

where $d_{max}$ is the graph's diameter. In practice, if $d_{max}$ is unknown for a large graph it can estimated, as long as the estimate is greater than $d_{max}$. We need this so that the measure is still able to predict $d_{max}$ despite the restriction of $\delta$ to $[-1, 1]$.

## III. Experiments

### A. Datasets

TABLE I
DESCRIPTIONS OF DATASETS USED IN OUR EXPERIMENTS

| | —V— | —E— | AVG DEG | $d_{max}$ |
|---|---|---|---|---|
| HARTFORD (**HF**) | 1581 | 2470 | 2.75 | 12 KM |
| SURAT (**SU**) | 2591 | 3670 | 2.86 | 51 KM |
| EGO-FACEBOOK (**FB**) | 4039 | 88234 | 2 | 8 |
| DONGGUAN (**DG**) | 8315 | 11128 | 2.32 | 160 KM |

We ran experiments on several graphs in order to compare our model to previous work. Hartford, Connecticut is a geographic driving network from OSMnx [13]. Surat, India and Dongguan, China are geographic networks from the CSUN lab [18], and ego-facebook-original is a social network graph from the SNAP dataset [19]. The Facebook graph is unweighted as opposed to all of the geographic graphs.

### B. Baselines

We compare our model against three previous deep learning architectures for distance approximation: Qi et al. [15] which introduces Vdist2vec, Rizi et al. [14], and Brunner [12]. We tailor our experimental setup for comparison purposes. We use $D = k = 128$ embeddings for all experiments except against Qi et al. We train Node2Vec with the recommended settings of walk length = 80, $p = q = 1$, 10 walks per node, 100 epochs, 256 batch size and lr=0.01, however we use a window size of 10 for Brunner and 5 for Rizi et al. since they differed in this respect. For Qi et al. we also use a window size of 10. For GraRep, the order was consistently set to 100. We also use 10 SVD iterations for all experiments and averaging to merge the $k$-step representations.

## IV. Results

Table II compares our best performing technique for each approximation experiment against the comparison models. We produce models which surpass two current approaches to distance approximation on geographic and social networks. Against the state-of-the-art approach for geographic networks, Vdist2vec, we achieve a superior error on one of our comparison graphs, and surpass their base model in MAE and MRE on the other graph. Of our models, Inv-Dot performed the best at distance approximation and routing consistently in the geographic networks we tested. However, it failed on the unweighted social network graph and the L6-Norm was most successful, as we discuss later.

### A. Experiment A: Vdist2vec Results

In Table III and Table IV, we show results against Vdist2vec on two geographic networks: **SU** and **DG** from [15]. We observe that at least in geographic networks, our Inv-Dot model is competitive with Vdist2vec but does not surpass it entirely in error as we will see with the other experiments. On **SU**, Inv-Dot performs between the initial Vdist2vec and Vdist2vec-S in terms of MRE regardless of initialization. When additionally trained on MAE, our model surpasses the base Vdist2vec entirely. On **DG**, our Inv-Dot with node2vec initialization model achieves state-of-the-art in terms of MRE, which occurs whether we train with MAE or MRE as the loss.

### B. Experiment B: Rizi et al. Results

In the non-geographic setting, we show a significant performance increase over the Rizi et al. model as evidenced by Table V showing approximation errors on the ego-facebook-original graph. Two of our models, elliptic and L6 norm with node2vec, show serious performance jumps relative to the Rizi et al. model. We display their result with the best (averaging, denoted $\oslash$) or second best (subtracting, denoted $\ominus$) performing input vector merging scheme for this graph. We should note that the elliptic measure result could be considered more significant since it was achieved after lowering, rather than raising the learning rate from the default for the experiment. Tweaking the learning rate down was helpful for most models, which is perhaps related to the low number of epochs set by Rizi et al. for this experiment.

TABLE II
EXPERIMENTS SUMMARY

| A - Surat | | | A - Dongguan | | | B - Facebook | | | C - Surat | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | **MAE** | **MRE** | **Method** | **MAE** | **MRE** | **Method** | **MAE** | **MRE** | **Method** | **MAE** | **MRE** |
| Inv-Dot n2v | 0.0098 | 0.027 | Inv-Dot n2v | 0.0196 | **0.01** | Elliptic | 0.0272 | 0.0157 | L6 | 0.0291 | 0.0357 |
| L6 n2v | 0.0194 | 0.0261 | | | | L6 n2v | **0.0267** | **0.0143** | Inv-Dot | **0.0161** | **0.0229** |
| Vdist2vec | 0.0113 | 0.027 | Vdist2vec | 0.0118 | 0.015 | Rizi ⊘ | 0.118 | 0.038 | CN | 0.0265 | 0.0378 |
| Vdist2vec-S | **0.0067** | **0.0114** | Vdist2vec-S | **0.0062** | 0.014 | Rizi ⊖ | 0.197 | 0.071 | SCN | 0.0238 | 0.0309 |

TABLE III
EXPERIMENT A - SURAT

| Technique | Initialization | Performance | | Params |
|---|---|---|---|---|
| | | **MAE** | **MRE** | |
| L6-Norm | Random | 0.0206 | 0.0266 | lr=0.001 |
| | Coord | 0.0328 | 0.034 | |
| | node2vec | 0.0194 | 0.0261 | |
| Inv-Dot | Random | 0.0161 | 0.0197 | |
| | Coord | 0.0163 | 0.0207 | |
| | node2vec | 0.0159 | 0.019 | |
| | GraRep | 0.0164 | 0.0209 | |
| Inv-Dot MAE | node2vec | 0.0109 | 0.0239 | |
| Inv-Dot MSE | node2vec | 0.0098 | 0.027 | lr=0.0003 |
| Vdist2vec base | N/A | 0.0113 | 0.027 | |
| Vdist2vec-S | | **0.0067** | **0.014** | |

TABLE IV
EXPERIMENT A - DONGGUAN

| Technique | Initialization | Performance | | Params |
|---|---|---|---|---|
| | | **MAE** | **MRE** | |
| L6-Norm | Random | 0.0355 | 0.0286 | lr=0.001 |
| | Coord | 0.0679 | 0.0374 | |
| | node2vec | 0.0349 | 0.0304 | |
| Inv-Dot | Random | 0.0207 | 0.0345 | |
| | Coord | 0.021 | 0.0121 | |
| | node2vec | 0.0196 | **0.01** | |
| Inv-Dot MAE | node2vec | 0.0178 | 0.0123 | |
| Inv-Dot MSE | node2vec | 0.0211 | 0.0171 | |
| Vdist2vec base | | 0.0118 | 0.015 | |
| Vdist2vec-S | | **0.0062** | 0.014 | |

TABLE V
EXPERIMENT B - FACEBOOK

| Technique | Initialization | Performance | | Params |
|---|---|---|---|---|
| | | **MAE** | **MRE** | |
| L2-Norm | Random | 0.114 | 0.0586 | lr=0.008 |
| L6-Norm | Random | 0.0482 | 0.022 | lr=0.001 |
| | node2vec | **0.0267** | **0.0143** | lr=0.03 |
| | GraRep | 0.1317 | 0.056 | |
| Elliptic | Random | 0.0272 | 0.0157 | lr=0.0003 |
| Poincare | Random | | | |
| Minkowski | Random | 0.4068 | 0.1464 | lr=0.005 |
| Inv-Dot | Random | 0.6278 | 0.1637 | lr=2e-5 |
| | node2vec | 0.5001 | 0.1365 | |
| Rizi | node2vec ⊘ | 0.118 | 0.038 | |
| Rizi | node2vec ⊖ | 0.197 | 0.071 | |

### C. Experiment C: Brunner Results

Against Brunner we find that with coordinate initialization several of our measures are competitive with their graph neural network architecture, as shown in Table VI. L4, L6, and L8 are all within the same range of performance as these networks, illustrating that rich representations can be learned even when the distance measure is not an ideal approximation with respect to the provided initial coordinates. The more typical L2 distance also performs in the range in MAE, even though we train on MRE. As expected, hyperbolic distance models do not perform as well as $p$-norms or dot product, which are more intuitive measures for graphs representing physical geometries. As with the previous experiment on geographic networks, our best performance comes from the Inv-Dot measure, which surpasses all comparison models on this experiment.

Since Qi et al. and Brunner both ran experiments on the **SU** graph, we can also get some sense of how the difference in experiment setup affects the performance of our methods. We found the Brunner setup to be much more stable generally and so required no hyperparameters tweaks for specific models. In comparison to the Qi experiment, it used a lower learning rate ($0.01 \rightarrow 0.001$), smaller batch size ( $2500 \rightarrow 512$) and more train epochs ($20 \rightarrow 100$). Most significantly, Qi et al. train on all node pairs, whereas the Brunner experiments use 10% train ratio and tested on the remainder. We can see that between these two experiments L6 and Inv-Dot setups performed similarly. This is good news since it suggests that the models achieve good representations without simply memorizing distances for the provided node pairs. In the future, experiments with lower train ratios and larger graphs may be more useful to evaluate performance, since this line of research in distance approximations with embeddings or neural networks is only useful when $|V|$ is too large to be able to generate the shortest path distances for all node pairs in a feasible amount of time with modest space available.

### D. Experiment D: Hartford Routing Results

Finally, we evaluate our methods on an $A^*$ routing task on the Hartford network, and show results in Table VII, in comparison to a baseline distance heuristic using the geographic coordinates. From our results, we can see that in comparison to the baseline geographic distance heuristic, all methods produce better average $Q$, but not to a very significant degree in most cases.

They also differ greatly in the performance distribution by which their average error is achieved. For instance, the Poincare Hyperbolic and Elliptic measures produce the second and third-best 99th percentile $Q$, but second and third-worst median $Q$. They may produce generally well-distributed embeddings but are unable to fit in error very closely. For this

TABLE VI
EXPERIMENT C - SURAT

| Technique | Performance | |
|---|---|---|
| | MAE | MRE |
| L2-Norm | 0.0411 | 0.0605 |
| L4-Norm | 0.0299 | 0.0385 |
| L6-Norm | 0.0291 | 0.0357 |
| L8-Norm | 0.0307 | 0.0372 |
| Poincare | 0.0951 | 0.0404 |
| Minkowski | 0.3525 | 0.2101 |
| Inv-Dot | **0.0161** | **0.0229** |
| CN | 0.0265 | 0.0378 |
| SCN | 0.0238 | 0.0309 |
| SECN | 0.0433 | 0.0450 |
| SACN | 0.0419 | 0.0436 |

reason, we think it is worth investigating how they could be redesigned so that they produce embeddings in hyperbolic and elliptic geometry, respectively, but with improved performance in model training. On the other hand, the L6 norm performed the second best in 50th and the worst in 99th percentile $Q$. Considering the intuition for performance of higher $p$-norms, this may be because the model is attempting to represent the graph adjacency matrix in the embedding space by ignoring some parts of the network, at least more so than the other methods. This would allow for better median performance at the expense of edge case performance.

## V. DISCUSSION

In general, we found on the tested graphs that higher norms had better output performance. To see potentially why this could be, suppose we have a 2-dimensional embedding matrix, MAE loss, and that $|\mathbf{u}_1 - \mathbf{v}_1| = d_{u,v}$ for a particular training pair $(u, v)$. Then we have the loss for this example as

$$L = \left| \left( (\mathbf{u}_1 - \mathbf{v}_1)^p + d_{u,v}^p \right)^{1/p} \right|$$

which is minimized for a larger range of $\mathbf{u}_1 - \mathbf{v}_1$ around 0 when $p$ is large. This means if the distance is already well approximated by some coordinate, the gradient update for the other coordinates will be small as long as the difference is not excessively large, so they can instead be tuned for other examples. Given this, the theoretical optimal embedding for the norm measure would be using the $\infty$-norm with $d$ as the maximum degree of $G$, and would have that for each pair $(u, v)$ there is a corresponding coordinate $i^*$ in the embedding such that $i^* = \text{argmax}_i |\mathbf{u}_i - \mathbf{v}_i| = d_{u,v}$. Such an embedding would be similar to an adjacency matrix for $G$. It is not obvious whether such an embedding is possible generally or for any $G$, so this question is something to pursue in future work. However, we can say that higher $p$-norms are most likely better able to approximate the adjacency matrix in the lower dimensional embedding space $|V| \times D$.

However, the model can only train for a certain number of iterations, and higher $p$ values result in lower initial prediction values. Given the experimental setups we use, it appears empirically that the L6-Norm is the optimal point in this tradeoff between poor initialization and better optima.

The other best performing measure was Inv-Dot as a measure was motivated by empirical issues with using the Elliptic measure on geographic networks. Ignoring scaling into the proper range $x \in [-1, 1]$ and $y \in [0, d_{max}]$ performed for both measures, the change is simply to replace $\cos^{-1}(x)$ with $-x$, which is a much simpler gradient and is constant as long as the input $x \in (-1, 1)$. Since this change is small, we hypothesize the performance in geographic networks is mainly due to the change in gradient which allows faster training. Further investigation is necessary to understand the interesting results on Ego Facebook and whether these results generalize to other social networks.

## VI. CONCLUSION

We've proposed a simple embedding approach, ANEDA, to learn distance preserving graph embeddings in explicit representation spaces, and demonstrated comparable or better performance against state-of-the-art deep learning architectures.

For future investigation, we consider the following areas of extension or improvement of our work:

1) Verify whether using specific measures is successful in most cases at embedding into the respective mathematical space (within some margin of error).
2) Modify measures for certain geometries (e.g. hyperbolic, elliptic) to produce better gradients.
3) Introduce a loss or training constraint to better enforce the triangle inequality so that our model defines a metric space.
4) Design a custom loss to target worst-case distance approximation, particularly for use in downstream routing tasks.
5) Apply the technique in an online learning setting, where embeddings must be updated as changes to the graph occur.

## REFERENCES


[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[2] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[3] A. Grover and J. Leskovec, *Node2vec: Scalable feature learning for networks*, 2016. arXiv: 1607.00653 [cs.SI].

TABLE VII
EXPERIMENT D - HARTFORD ROUTING
($Q$ = PERCENT UNNECESSARY VISITS)

| Technique | Initialization | Q-Average | Q-50th | Q-90th | Q-99th | Routing Time |
|---|---|---|---|---|---|---|
| L2-Norm | Coord | 14.45 | 8.7 | 37.5 | 86.06 | **18:25** |
| L6-Norm | Coord | 20.09 | 8.33 | 67.86 | 97.14 | 22:21 |
| | node2vec | 17.12 | 7.55 | 54.05 | 96.16 | 21:58 |
| Inv-Dot | Coord | 22.55 | 13.89 | 60.98 | 89.77 | 20:27 |
| | node2vec | **12.53** | **7.14** | **32.56** | **82.59** | 18:44 |
| Poincare | Coord | 15.86 | 9.76 | 40.32 | 86.23 | 18:32 |
| Spherical | Coord | 15.33 | 9.43 | 38.36 | 85.71 | 19:01 |
| Geographic Dist | | 23.86 | 8.33 | 54.29 | 96.81 | 22:54 |


[4] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "Struc2vec," *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2017. DOI: 10.1145/3097983.3098061. [Online]. Available: http://dx.doi.org/10.1145/3097983.3098061.

[5] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, *Don't walk, skip! online learning of multi-scale network embeddings*, 2017. arXiv: 1605.02115 [cs.SI].

[6] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, *Harp: Hierarchical representation learning for networks*, 2017. arXiv: 1706.07845 [cs.SI].

[7] J. Schlötterer, M. Wehking, F. S. Rizi, and M. Granitzer, "Investigating extensions to random walk based graph embedding," *2019 IEEE International Conference on Cognitive Computing (ICCC)*, pp. 81–89, 2019.

[8] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW '13, Rio de Janeiro, Brazil: Association for Computing Machinery, 2013, pp. 37–48, ISBN: 9781450320351. DOI: 10.1145/2488388.2488393. [Online]. Available: https://doi.org/10.1145/2488388.2488393.

[9] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, 2015, pp. 891–900.

[10] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1105–1114, ISBN: 9781450342322. DOI: 10.1145/2939672.2939751. [Online]. Available: https://doi.org/10.1145/2939672.2939751.

[11] J. Zhang, Y. Dong, Y. Wang, J. Tang, and M. Ding, "Prone: Fast and scalable network representation learning," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization, Jul. 2019, pp. 4278–4284. DOI: 10.24963/ijcai.2019/594. [Online]. Available: https://doi.org/10.24963/ijcai.2019/594.

[12] D. Brunner, "Distance preserving graph embedding," 2021.

[13] G. Boeing, "Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, vol. 65, 2017, pp. 126–139.

[14] F. S. Rizi, J. Schloetterer, and M. Granitzer, "Shortest path distance approximation using deep learning techniques," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, IEEE, 2018, pp. 1007–1014.

[15] J. Qi, W. Wang, R. Zhang, and Z. Zhao, "A learning based approach to predict shortest-path distances," in *EDBT*, 2020.

[16] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 855–864, ISBN: 9781450342322. DOI: 10.1145/2939672.2939754. [Online]. Available: https://doi.org/10.1145/2939672.2939754.

[17] P. Tabaghi and I. Dokmanić, "Hyperbolic distance matrices," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1728–1738.

[18] A. Karduni, A. Kermanshah, and S. Derrible, "A protocol to convert spatial polyline data to network formats and applications to world urban road networks," 160046, vol. 3, Scientific Data, 2016. [Online]. Available: https://doi.org/10.6084/m9.figshare.2061897.v1.

[19] J. Leskovec and A. Krevl, *Snap datasets: Stanford large network dataset collection*, 2014.