Fast link prediction for large networks using spectral embedding

BENJAMIN PACHEV, BENJAMIN WEBB*
Department of Mathematics, Brigham Young University, Provo, Utah
*Corresponding author. Email: bwebb@mathematics.byu.edu

[Received on 9 April 2017]

Many link prediction algorithms require the computation of a similarity metric on each vertex pair, which is quadratic in the number of vertices and infeasible for large networks. We develop a class of link prediction algorithms based on a spectral embedding and the k closest pairs algorithm that are scalable to very large networks. We compare the prediction accuracy and runtime of these methods to existing algorithms on several large link prediction tasks. Our methods achieve comparable accuracy to standard algorithms but are significantly faster.

Keywords: link prediction; graph embedding; commute time; resistance distance; closest pairs.

1. Introduction

The study of networks has become increasingly relevant in our understanding of the technological, natural, and social sciences. This is owing to the fact that many important systems in these areas can be described in terms of networks [1], where vertices represent the system's individual components, e.g. computer routers, neurons, individuals, etc. and where edges represent interactions or relationships between these components.

An essential feature of the large majority of these networks is that they have a dynamic topology, i.e. a structure of interactions that evolves over time [2]. The structure of social networks, for instance, change over time as relationships are formed and dissolved. In information networks such as the WWW the network's structure changes as information is created, updated, and linked.

Although understanding the mechanisms that govern this structural evolution is fundamental to network science, these mechanisms are still poorly understood. Consequently, predicting a network's eventual structure, function, or whether the network is likely to fail at some point are all currently out of reach for even simple networks.

In an attempt to determine which processes cause changes in a network's structure we are lead to the following link prediction problem: Given a network, which of the *links*, i.e. edges between existing vertices, are likely to form in the near future. Here we adopt the standard convention that links are to be predicted solely on the basis of the network's current topology (see, for instance, [3]).

Importantly, the link prediction problem can be used to study more than just which edges will appear in a network. It can also be used to predict which of the non-network edges are, in fact, in the network but currently undetected. Similarly, it can be used to detect which of the current network edges have been falsely determined to be a part of the network.

This notion of link prediction is of central importance in numerous applications. Companies such as Facebook, Twitter, and Google need to know the current state and efficiently predict the future structure of the networks they use to accurately sort and organize data [4]. Biologists need to know whether biochemical reactions are caused by specific sets of enzymes to infer causality and so on [5].

The barrier in determining whether network links truly exist in these and other settings, is that testing and discovering interactions in a network requires significant experimental effort in the laboratory or in the field [6]. Similarly, determining experimentally when and where a new link will form may also be impractical, especially if the precise mechanism for link formation is unknown. For these reasons it is important to develop models for link prediction.

At present, there is an ever increasing number of proposed methods for predicting network links [7]. Not surprisingly, certain methods more accurately predict the formation of links in certain networks when compared with others. Additionally, each of these methods has a runtime that scales differently with the size of the network. In our experiments, we discover that a number of link predictors have a runtime that is so high that it effectively prohibits their use on moderately large networks.

Here we propose a class of link predicting algorithms that scale to large networks. This method, which we refer to as the *approximate resistance distance predictor*, integrates a spectral embedding of the network with a known algorithm for efficiently finding the k closest pairs of points in Euclidean space. The spectral embedding aspect of the algorithm is derived as a low-rank approximation of the effective resistance between network vertices, as in [8]. The k closest pairs component of the algorithm is taken from [9] and can be used to predict links based on this embedding.

Here we compare the prediction accuracy and runtime of this method against several well-known algorithms on a number of coauthorship networks and a social network consisting of a small subset of Facebook users. We find that our method is achieves the best accuracy on some networks and scales to networks that many other link predictors cannot.

The paper is structured as follows. In Section 2 we describe the link prediction problem and outline a number of standard link prediction algorithms. In Section 3 we introduce the method of resistance distance embedding and prove that it is optimal as a low rank approximation of effective resistance (see Proposition 3.1). In Section 4 we describe the experimental setup. Section 5 numerical results comparing the performance of the resistance distance embedding algorithm to other algorithms are given. Section VI concludes with some closing remarks including a number of open questions for future work.

2. The Link Prediction Problem

The link prediction problem can be stated as follows. Given a connected graph $G = (V,E)$, and k, the number of predicted nonadjacent links, we seek k pairs of vertices which are most likely to become connected. While the choice of k depends on the application, we adopt the convention that $1 \leqslant k \leqslant |E|$.

The general paradigm for link prediction is to compute a similarity metric $score(x,y)$ on each vertex pair (x,y). The predicted links are then the k $(x,y) \in V \times V - E$ for which $score(x,y)$ is maximal. By contructing a matrix from the scores, we obtain a *graph kernel*. We can also go in the other direction. Any real $n \times n$ matrix, where $n = |V|$, defines a score function on pairs of vertices, and can be used for link prediction.

We now give a sampling of existing link prediction algorithms.

2.1 *Local Methods*

A *local method* for link prediction is an algorithm that uses vertex neighborhoods to compute similarity.

Common Neighbors: Common neighbors simply assigns

$$score(x,y) = |\Gamma(x) \cap \Gamma(y)|, \tag{2.1}$$

where $\Gamma(x)$ is the neighbor set for $x \in V$.

Jaccard's Coefficient: Jaccard's coefficient is a normalized version of common neighbors that takes into account the total number of neighbors for both vertices. It is given by

$$score(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}. \tag{2.2}$$

Preferential Attachment: Preferential attachement is based on the idea that highly connected nodes are more likely to form links, an observed pattern in coathourship networks [10]. This leads to

$$score(x,y) = |\Gamma(x)||\Gamma(y)|. \tag{2.3}$$

Adamic-Adar:

$$score(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{log|\Gamma(z)|} \tag{2.4}$$

Resource Allocation:

$$score(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{|\Gamma(z)|} \tag{2.5}$$

2.2 *Path-based Methods*

Path-based methods consider all or a subset of the paths between two vertices to compute similarity. Unlike local similarity measures, they can capture global information about the network.

Shortest Path: This link predictor defines $score(x,y)$ as the negated length of the shortest path from x to y.

Katz: The Katz metric counts all paths between two nodes, and discounts the longer paths exponentially. Define $path_{x,y}^{\ell}$ to be the set of all paths of length ℓ from x to y. Then given a weight $0 < \beta < 1$,

$$score(x,y) = \sum_{\ell=1}^{\infty} \beta^{\ell} |path_{x,y}^{\ell}| \tag{2.6}$$

A closed form for the associated graph kernel is given by $(I - \beta A)^{-1} - I = \sum_{\ell=1}^{\infty} (\beta A)^{\ell}$, where A is the adjacency matrix of G.

2.3 *Random walks*

A *random walk* on G starts at some node x and iteratively moves to new nodes with uniform probability. There are a multitude of link predictors based on random walks. These are some of the fundamental ones.

Hitting and Commute Time: The *hitting time* $H_{x,y}$ is the expected number of steps required to reach y in a random walk starting at x. Commute time is defined as $C_{x,y} = H_{x,y} + H_{y,x}$. Negated hitting time can be used as a link predictor, but the hitting time is assymetric in general, so we use instead the negated commute time, which is symmetric.

The commute time and its variants will be discussed further in Section 3.

Rooted Page Rank: A problem with hitting and commute time is that random walks can become lost exploring distant portions of the graph. Rooted Page Rank deals with this problem by introducing random resets. Given a root node x, we consider a random walk starting at x. At each step, with probability α the walk returns back to x. With probability $1 - \alpha$ the walk proceeds to a random neighbor. Given a root node x, for each other node y, $score(x,y)$ is defined as the stationary probability of y under the random walk rooted at x. The corresponding graph kernel is given by $(1-\alpha)(I-\alpha D^{-1}A)^{-1}$, where D is the degree matrix and A is the adjacency matrix.

2.4 *Scaling Link Predictors to Large Networks*

Many link predictors, such as Katz, require the computation of a matrix inverse. This is heinously expensive for large networks, as it is cubic in the number of vertices. One way to circumvent such problems is via a low-rank approximation of the score matrix. We investigate such a low-rank approximation for the commute-time or resistance distance kernel in the next section.

Even the simpler local predictors such as common neighbors or preferential attachment face difficulties at scale. This is because for sufficiently large networks, it is not possible to compute scores for each pair of vertices and then find the maximal ones. Instead, efficient search techniques must be employed to search only a small subset of the potential links in order to find those of maximal score. In Section 3 we will demonstrate how a class of graph embedding based predictors can efficiently find the k links of maximal score.

3. Spectral Embedding

We begin by deriving the *approximate resistance distance link predictor* as a best low-rank approximation to commute time and show how to evaluate its link prediction scores with a spectral embedding. We then show that this link predictor is part of a family of graph embedding based link predictors that use the k closest pairs algorithm to efficiently find the links of maximal score. Finally, we discuss efficient ways to compute the spectral embedding upon which the approximate resistance distance predictor relies.

3.1 *Approximating Commute Time*

Let $L = D - A$ be the Laplacian matrix of a graph $G = (V,E)$, and let $n = |V|$. Let $L^\dagger$ be the Moore-Penrose inverse of L. Then the commute time is given by

$$C_{x,y} = |E|(L^\dagger_{x,x} + L^\dagger_{y,y} - 2L^\dagger_{x,y}), \tag{3.1}$$

where the quantity $r_{x,y} = (L^\dagger_{x,x} + L^\dagger_{y,y} - 2L^\dagger_{x,y})$ is known as the *effective resistance* or the *resistance distance* [8]. Since resistance distance differs from commute-time by a (network-dependant) constant scaling factor, they can be used interchangeably for link prediction.

For many networks, G is too large to compute $L^\dagger$ exactly, so an approximation must be used. A natural choice is a best rank-d approximation to $L^\dagger$ for some fixed dimension d. The resulting approximation of the resistance distances is closely related to distances between points in Euclidean space.

PROPOSITION 3.1 Let d be a positive integer and let $G = (V,E)$ be a connected, undirected graph. Then $\exists$ a best rank-d approximation S of $L^\dagger$, and a map $f : V \to R^d$ so that $\forall\ x,y \in V$, $S_{x,x} + S_{y,y} - 2S_{x,y} = \|f(x) - f(y)\|_2^2$. We call this map the *resistance distance embedding*.

Proof. For a connected graph, the Laplacian matrix is positive semidefinite, with eigenvalues $0 = \lambda_1 < \lambda_2 \leqslant \cdots \leqslant \lambda_n$ and corresponding eigenvectors $v_1, v_2, v_3, \ldots, v_n$. Then we have the spectral decompositions

$$L = \sum_{i=2}^{n} \lambda_i v_i v_i^T$$

and

$$L^{\dagger} = \sum_{i=2}^{n} \frac{1}{\lambda_i} v_i v_i^T.$$

Hence, $S = \sum_{i=2}^{d+1} \frac{1}{\lambda_i} v_i v_i^T$ is a best rank-d approximation to $L^{\dagger}$ in the 2-norm. Then note

$$\begin{aligned} S_{x,x} + S_{y,y} - 2S_{x,y} &= (e_x - e_y)^T S (e_x - e_y) \\ &= \sum_{i=2}^{d+1} \frac{1}{\lambda_i} (e_x - e_y)^T v_i v_i^T (e_x - e_y) \\ &= \sum_{i=2}^{d+1} \frac{1}{\lambda_i} (v_{i,x} - v_{i,y})^2 \\ &= \| f(x) - f(y) \|_2^2 \end{aligned}$$

where

$$f(x) = [\frac{v_{2,x}}{\sqrt{\lambda_2}}, \frac{v_{3,x}}{\sqrt{\lambda_3}}, \ldots, \frac{v_{d+1,x}}{\sqrt{\lambda_{d+1}}}]^T \in R^d \tag{3.2}$$

□

We define the *approximate resistance distance link predictor* of dimension d by setting

$$score(x,y) = -(S_{x,x} + S_{y,y} - 2S_{x,y}) = -\| f(x) - f(y) \|_2^2, \tag{3.3}$$

where S and f are defined as in Proposition 3.1.

In the next section, we will see that the approximate resistance distance link predictor is part of a class of link predictors that avoid brute-force search when predicting links.

3.2 *Link Prediction with Graph Embeddings*

The resistance distance embedding is a special case of a *graph embedding*, which is a map f from V to R^d, d a positive integer. We can use graph embeddings to create link predictors. A natural choice is to set $score(x,y) = -\| f(x) - f(y) \|_2$, (so maximizing score corresponds to minimizing distance). We refer to this score function as the *Euclidean score*.

If f is the resistance distance embedding, then link prediction with the Euclidean score is equivalent to the approximate resistance distance predictor. Recall that the approximate resistance distance score function is $-\| f(x) - f(y) \|_2^2$. The k predicted links of maximal score correspond to the k nonadjacent pairs of vertices (x,y) for which $-\| f(x) - f(y) \|_2^2$ is maximal. These are precisely the k links for which $\| f(x) - f(y) \|_2$ is minimal and are predicted with the Euclidean score.

Link prediction with the Euclidean score is related to the k *closest pairs problem*. The closest pairs problem is as follows. Given a set of vectors $\{x_1, x_2, \ldots, x_n\} \subset R^d$ we seek the k unordered pairs $(x_i, x_j), i \neq j$ of minimal distance (here we use the Euclidean norm but any L_p norm can be used, $1 \leqslant p \leqslant \infty$). There is an algorithm to solve this problem in

$$O(d(n\,log\,n + k\,log\,n\,log(\frac{n^2}{k}))) \tag{3.4}$$

[9].

We can think of the link prediction problem as the closest pairs problem applied to the set of vectors $\{f(y), y \in V\}$, with the additional constraint that the best pairs must correspond to non-edges in G. The extra constraint can be handled by finding the $|E|+k$ closest pairs, then selecting the best k which are non-edges. As there can be no more than $|E|$ edges, this approach is sure to work. We then have the worst-case complexity bound of

$$O(d(n\,log\,n + (|E|+k)log\,n\,log(\frac{n^2}{|E|+k}))). \tag{3.5}$$

Recalling that we require $1 \leqslant k \leqslant |E|$, and assuming that G is connected so $|E| \geqslant n-1$, this complexity bound can be simplified to

$$O(d\,|E|\,log^2 n). \tag{3.6}$$

For large, sparse networks, $|E| << n^2$, and this is a tremendous speedup over the $O(n^2)$ brute-force approach.

Cosine Similarity Score: Another link prediction score function that can be derived from a graph embedding is the cosine similarity score, defined by

$$score(x,y) = \frac{< f(x), f(y) >}{\|f(x)\|\,\|f(y)\|}. \tag{3.7}$$

If the cosine similarity score is used, the link prediction problem can still be solved without brute-force search. It is equivalent to the link prediction problem with Euclidean score on a modified graph embedding. The modified embedding is obtained from the original by normalizing the embedding vectors as follows.

PROPOSITION 3.2 Given a graph embedding $f : V \to R^d$, the link prediction problem using

$$score(x,y) = \frac{< f(x), (f(y)) >}{\|f(x)\|\,\|f(y)\|} = \cos\theta$$

is equivalent to the link prediction problem with the Euclidean score function on the modified embedding given by $g(y) = \frac{f(y)}{\|f(y)\|}$.

Proof. Let $x, y \in V$. Note

$$< g(x), g(y) > = \cos\theta = score(x,y).$$

We have

$$\|g(x)-g(y)\|_2^2 = \|g(x)\|_2^2 + \|g(y)\|_2^2 - 2 < g(x), g(y) >$$
$$= 2 - 2\cos\theta = 2 - 2score(x,y).$$

This shows that minimizing Euclidean distance for the modified embedding is the same as maximizing cosine similarity score on the original, so link prediction with Euclidean score on the modified embedding is equivalent to link prediction with the cosine similarity score on the original. □

This section introduced a class of link predictors that avoid a brute-force search when predicting links. These link predictors rely on a precomputed graph embedding. The graph embedding needs to be efficiently computable in order for the overall prediction algorithm to be fast. We are concerned with link predictors that rely on the resistance distance embedding. Consequently, rapid computation of this particular graph embedding is the subject of the next section.

3.3 *Computing the Resistance Distance Embedding*

Computing the resistance distance embedding of dimension d requires finding the smallest d nonzero eigenvalues and associated eigenvectors of the Laplacian matrix L. Fortunately, specialized, efficient algorithms exist for this problem which exploit the positive semi-definiteness and sparsity of L. These include TRACEMIN-Fiedler [11] and a multilevel solver MC73_FIEDLER [12]. TRACEMIN-Fiedler is simpler to implement, and is also parallelizable, so we use it in our experiments.

4. Experimental Setup

In this section we compare the performance of our link prediction algorithm to others on several large social networks. In a social network, nodes correspond to persons or entities. Edges correspond to an interaction between nodes, such as coauthouring a paper or becoming linked on a social media website.

4.1 *The Networks*

Arxiv High Energy Physics Theory (hep-th): This network is a coauthorship network obtained from the Konect network collection. [13, 14].

Arxiv High Enery Physics Phenomenology (hep-ph): This is another coauthorship network from the Konect network collection [13, 15].

Facebook Friendship (facebook): This social network consists of a small subset of facebook users, where edges represent friendships [16, 17].

Arxiv Condensed Matter Physics (cond-mat): This dataset was obtained from Mark Newman's website [18], and is also a coathourship network. Unlike the other datasets, the edges are not timestamped.

4.2 *Creating Training Graphs*

In order to perform link prediction, we partition edges into a training set and a test set. Edges in the training set occur before those in the test set and are used to construct a training graph. We run link prediction algorithms on the training graph to predict the contents of the test set. In most cases, edges have timestamps, and we can choose a cutoff time to partition the edges.

For one network (cond-mat) the edges are not timestamped. However, there are two versions of the cond-mat network available. One contains all collaborations up to 2003. The second is an updated network with all collaborataions up to 2005. We use the first network as the training graph. The test set consists of all edges in the second network for which both nodes are in the earlier network.

Choosing the cutoff between the training and test edges is somewhat arbitrary. If too few edges are used for training, link predictors will struggle. If too few are left for testing, then results may be statistically insignificant. See Table 1 for a comparison of the training networks and original networks.

Our spectral embedding based link prediction algorithms require a connected graph. To solve this problem, we reduce each training graph to its largest connected component. For each network we consider, the largest component contains the vast majority of the vertices.

4.3 *The Predictors*

We perform experiments with two spectral embedding based predictors. Each uses the resistance distance embedding of dimension d, with d a parameter to be varied. The first uses the Euclidean score function and is equivalent to the approximate resistance distance predictor of dimension d. The second

Network	Nodes	Edges	Average Degree
cond-mat	15,803	60,989	7.7187
cond-mat train	13,861	44,619	6.4381
facebook	63,731	817,035	12.8201
facebook train	59,416	731,929	24.6374
hep-ph	28,093	3,148,447	112.0723
hep-ph train	26,738	2,114,734	158.1819
hep-th	22,908	2,444,798	106.7225
hep-th train	21,178	1,787,157	168.7749

Table 1. Training network statistics

uses the cosine similarity score. We refer to these link predictors as spec_euclid and spec_cosine respectively (spec for spectral). In tables, the dimension of the embedding is indicated by a number after the predictor name. For example, spec_euclid8 refers to the spec_euclid predictor using an 8-dimensional resistance distance embedding.

The other link prediction algorithms used in our experiments are preferential attachment, common neighbors, Adamic Adar, Rooted Page Rank and Katz (with $\beta = .01$). Some networks are too large for certain algorithms to handle, so not every algorithm is run on each network. For example, the facebook training graph has 59,416 nodes. Computing the Katz score on this graph requires finding the inverse of a 59,416$\times$59,416 matrix, and is very expensive in time and space, so we do not use the Katz algorithm for the facebook graph.

All experiments were performed on the same 4 core machine. The common neighbors, preferential attachment, and Adamic Adar algorithms were implemented in Python and were not parallelized. Our spectral link predictors, Katz, and Rooted Page Rank use the Python library Numpy to parallelize linear algebra operations. All code that was used in the experments in this paper can be found at the git repository bitbucket.org/thorfax/spectral_research.

For each network, we fix the number of links to be predicted. With the exception of hep-th, this number is equal to 10% of the maximum possible number of correct predictions (i.e the number of new links in the test set). For the hep-th network we discovered that the spec_euclid and spec_cosine predictors achieve nearly perfect accuracy when predicting 1000 links. As this is not the case for any other network we considered, we report this unusual phenomenon.

For all of the networks we consider, the probability of randomly predicting a correct link is very low. Most of the algorithms we consider do much better than the random baseline, but have low raw accuracy since there are few new links compared to the number of possible links. See Table 2 for a summary of the number of links predicted and baseline probability of randomly predicting a correct link.

5. Results

On the cond-mat and facebook networks, both the spec_euclid and spec_cosine predictors performed worse than the simple common neighbors predictor. In addition to the full networks, we also compared predictor accuracy on reduced versions of the hep-th and hep-ph networks, because the full networks are too large for methods like Katz, common neighbors, and Rooted Page Rank to complete in a reasonable amount of time. On our reduced version of the hep-th network, our embedding-based predictors did better than common neighbors but not as well as the Rooted Page Rank predictor. On the reduced hep-

Network	Links	Random Accuracy (%)
cond-mat	1190	0.012
facebook	7858	0.004
hep-ph	101466	0.286
reduced hep-ph	1988	0.661
hep-th	1000	0.296
reduced hep-th	135	0.084

Table 2. Link Prediction Task Setup

ph network, the spec_euclid predictor performed significantly better than all other competitors, including our other embedding-based predictor, spec_cosine.

As Table 3 shows, the best predictors for the cond-mat network were Katz and common neighbors. Note that for both spec_euclid and spec_cosine, the accuracy increases with the dimension of the embedding.

Predictor	Correct (%)	Time (s)
katz	5.97	62.96
commonNeighbors	5.97	1.55
prefattach	1.93	0.35
spec_euclid1	1.51	2.99
spec_cosine1	0.25	3.35
spec_euclid2	1.51	3.65
spec_cosine2	1.18	3.80
spec_euclid4	1.76	10.54
spec_cosine4	1.34	10.89
spec_euclid8	1.68	11.41
spec_cosine8	1.34	10.73
spec_euclid16	1.68	29.91
spec_cosine16	1.43	32.31

Table 3. Performance of link predictors on the cond-mat network

As previously mentioned, the facebook graph was too large to run the Katz predictor on it in a reasonable amount of time. As with the cond-mat network, the simple common neighbors predictor performs best.

Our spectral embedding link predictors performed significantly better on the hep-th and hep-ph networks, as Table 5 and Table 6 show.

The common neighbors algorithm did not scale to the hep-th and hep-ph networks, unlike the facebook network. Although the facebook network had more nodes, it has a lower average node degree and fewer distance two pairs. The common neighbors algorithm computes intersections of neighbor sets for each distance two pair. Because the average node degree is higher for the hep-th and hep-ph networks, these intersections are more expensive to compute and there are more distance two pairs for which intersections must be computed.

In order to compare the performance of our spectral predictors to other predictors on the hep-ph and

Predictor	Correct (%)	Time (s)
commonNeighbors	5.29	151.76
prefattach	0.41	7.00
spec_euclid1	0.42	9.86
spec_cosine1	0.00	47.20
spec_euclid2	0.50	11.52
spec_cosine2	0.42	12.96
spec_euclid4	1.40	24.05
spec_cosine4	1.02	25.96
spec_euclid8	1.95	61.21
spec_cosine8	2.58	62.27

Table 4. Performance of link predictors on the facebook network

Predictor	Correct (%)	Time (s)
prefattach	0.00	9.47
spec_euclid1	94.50	10.88
spec_cosine1	1.50	17.17
spec_euclid2	98.70	17.88
spec_cosine2	99.60	15.97
spec_cosine4	100.00	15.67
spec_euclid4	99.90	18.53
spec_euclid8	100.00	29.81
spec_cosine8	100.00	29.55
spec_euclid16	99.90	96.47
spec_cosine16	99.90	100.93

Table 5. Performance of link predictors on the hep-th network

Predictor	Correct (%)	Time (s)
prefattach	0.00	16.85
spec_euclid1	3.93	18.25
spec_cosine1	0.14	32.37
spec_euclid2	9.16	21.98
spec_cosine2	3.44	23.46
spec_euclid4	19.25	27.04
spec_cosine4	13.65	26.09
spec_euclid8	22.90	46.69
spec_cosine8	21.12	49.62
spec_euclid16	24.62	148.51
spec_cosine16	23.97	135.83

Table 6. Performance of link predictors on the hep-ph network

hep-th network data, we conducted another experiment using downsampled versions of these networks. To downsample, we used only the top 10% highest degree nodes. Our spectral predictors performed the best on the reduced hep-ph network (see Table 7), while the Rooted Page Rank algorithm was best for the reduced hep-th network (see Table 8).

Predictor	Correct (%)	Time (s)
prefattach	0.00	1.52
katz	1.16	2.75
commonNeighbors	9.36	23.96
pageRank	11.87	2.94
adamicAdar	8.85	754.40
spec_euclid1	1.81	2.80
spec_cosine1	0.10	6.21
spec_euclid2	4.73	3.73
spec_cosine2	2.57	6.86
spec_euclid4	13.13	5.80
spec_cosine4	11.12	9.01
spec_euclid8	16.40	7.39
spec_cosine8	9.31	7.44
spec_euclid16	14.13	17.32
spec_cosine16	4.93	14.90

Table 7. Performance of link predictors on the reduced hep-ph network

Predictor	Correct (%)	Time (s)
prefattach	0.00	1.28
katz	0.00	1.61
commonNeighbors	2.22	16.70
pageRank	11.11	1.97
adamicAdar	2.22	788.11
spec_euclid1	0.00	2.02
spec_cosine1	0.00	3.77
spec_euclid2	0.74	4.45
spec_cosine2	0.00	4.61
spec_euclid4	0.74	6.84
spec_cosine4	0.00	6.25
spec_euclid8	2.22	10.59
spec_cosine8	1.48	11.02
spec_euclid16	8.89	26.82
spec_cosine16	5.93	24.92

Table 8. Performance of link predictors on the reduced hep-th network

6. Conclusion

We present a link prediction framework that can scale to very large networks by avoiding the quadratic costs inherent in methods that exhaustively search all candidate pairs of nonadjacent nodes. We investigated the performance of a set of predictors based on this framework and the spectrum and eigenvectors of the graph's Laplacian matrix. These methods achieved high levels of accuracy on certain real-world link prediction tasks, and scaled well to networks with tens of thousands of nodes and millions of edges.

We emphasize that there are many other possible graph embeddings to invesitigate. Virtually all the runtime of our spectral link predictors is spent computing the resistance distance embedding. The k closest pairs component of our algorithm is very fast in practice, with nearly linear temporal complexity in the number of edges Replacing the resistance distance embedding with one that is cheaper to compute could potentially produce link predictors that can scale to much larger networks than the ones we consider in this paper.

Our approximate resistance distance link predictor was derived as a low-rank approximation of resistance distance, an established link prediction score that is expensive to compute. Many other well-known predictors are expensive to compute, such as Katz and Rooted Page Rank. There is much room to explore low-rank approximations of such predictors and investigate whether they can be converted into accurate, scalable, graph embedding based, link predictors of the form we considered.

Funding
This work was supported by the Defense Threat Reduction Agency [grant number HDTRA1-15-1-0049].

References

[1] Newman, M. (2010a) *Networks: An Introduction*. Oxford University Press.

[2] Gross, T. & Sayama, H. (2000b) *Adaptive Networks: Theory, Models and Applications*. Springer Publishing Company.

[3] Liben-Nowell, D. & Kleinberg, J. (2001c) The Link-Prediction Problem for Social Networks. *Journal of the American Society for Information Science and Technology*, **58**(7), 1019–1031.

[4] Quercia, D., Askham, H. & Crowcroft, J. (2012d) TweetLDA: supervised topic classification and link prediction in Twitter. in *Proceedings of the 4th Annual ACM Web Science Conference*, pp. 247–250. ACM.

[5] Barzel, B. & Barabási, A.-L. (2013e) Network link prediction by global silencing of indirect correlations. *Nature biotechnology*, **31**(8), 720–725.

[6] Clauset, A., Moore, C. & Newman, M. E. (2008f) Hierarchical structure and the prediction of missing links in networks. *Nature*, **453**(7191), 98–101.

[7] Srinivas, V. & Mitra, P. (2016g) *Link Prediction in Social Networks-Role of Power Law Distribution*. Springer.

[8] Fouss, F., Pirotte, A., Renders, J.-M. & Saerens, M. (2007h) Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on knowledge and data engineering*, **19**(3).

[9] LENHOF, H.-P. (1992i) The k closest pairs problem. http://people.scs.carleton.ca/∼michiel/k-closestnote.pdf, Accessed: 2017-03-11.

[10] NEWMAN, M. E. (2001j) Clustering and preferential attachment in growing networks. *Physical review E*, **64**(2), 025102.

[11] MANGUOGLU, M., COX, E., SAIED, F. & SAMEH, A. (2010k) TRACEMIN-Fiedler: A parallel algorithm for computing the Fiedler vector. in *International Conference on High Performance Computing for Computational Science*, pp. 449–455. Springer.

[12] HU, Y. & SCOTT, J. (2003l) *HSL MC73: A fast multilevel Fiedler and profile reduction code*. Rutherford Appleton Laboratory.

[13] LESKOVEC, J., KLEINBERG, J. & FALOUTSOS, C. (2007m) Graph Evolution: Densification and Shrinking Diameters. *ACM Trans. Knowledge Discovery from Data*, **1**(1), 1–40.

[14] NETWORKS COLLECTION, K. (2016n) arXiv hep-th network dataset – KONECT. http://konect.uni-koblenz.de/networks/ca-cit-HepTh, Accessed: 2017-03-11.

[15] ——— (2016o) arXiv hep-ph network dataset – KONECT. http://konect.uni-koblenz.de/networks/ca-cit-HepPh, Acessed: 2017-03-11.

[16] ——— (2016p) Facebook friendships network dataset – KONECT. http://konect.uni-koblenz.de/networks/facebook-wosn-links, Accessed: 2017-03-11.

[17] VISWANATH, B., MISLOVE, A., CHA, M. & GUMMADI, K. P. (2009q) On the Evolution of User Interaction in Facebook. in *Proc. Workshop on Online Social Networks*, pp. 37–42.

[18] NEWMAN, M. E. (2001r) The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, **98**(2), 404–409.