Social Network Analysis and Mining 2020 10:63
https://doi.org/10.1007/s13278-020-00671-6

ORIGINAL ARTICLE

Neighborhood and PageRank methods for pairwise link prediction

Huda Nassar[1] · **Austin R. Benson**[2] · **David F. Gleich**[3]

Received: 23 December 2019 / Revised: 23 June 2020 / Accepted: 23 June 2020

Abstract
Link prediction is a common problem in network science that cuts across many disciplines. The goal is to forecast the appearance of new links or to find links missing in the network. Typical methods for link prediction use the topology of the network to predict the most likely future or missing connections between a pair of nodes. However, network evolution is often mediated by higher-order structures involving more than pairs of nodes; for example, cliques on three nodes (also called triangles) are key to the structure of social networks, but the standard link prediction framework does not directly predict these structures. To address this gap, in recent work, we propose a new link prediction task called "pairwise link prediction" that directly targets the prediction of new triangles, where one is tasked with finding which nodes are most likely to form a triangle with a given edge. We extend this work in this manuscript, and we evaluate a variety of natural extensions to link prediction methods including neighborhood and PageRank-based methods. A key difference from our previous work is the definition of the neighborhood of an edge, which has a surprisingly large impact on the empirical performance. Our experiments on a variety of networks show that diffusion-based methods are less sensitive to the type of graphs used and more consistent in their results. We also show how our pairwise link prediction framework can be used to get better predictions within the context of standard link prediction evaluation.

Keywords Link prediction · Higher-order methods · PageRank · Neighborhood methods

1 Introduction

Networks are a standard tool for data analysis in which links between data points are the primary object of study. A fundamental problem in network analysis is *link prediction* (Liben-Nowell and Kleinberg 2007; Lü and Zhou 2011a), which is typically formulated as a problem of identifying pairs of nodes that will either form a link in the future (when viewing the network as evolving over time) or whose connection is missing from the data (Clauset et al. 2008). The link prediction problem has applications in a variety of domains. For instance, in online social networks of friendships, predicting that two people will form a connection can be used for friendship recommendation (Backstrom and Leskovec 2011). Similarly, predicting new links between users and items on platforms such as Amazon and Netflix can be used for product recommendation (Gomez-Uribe and Hunt 2015). And in biology, link prediction is used to identify novel interactions between genes, diseases, and drugs within interaction networks (Lin et al. 2018). In the settings above, the link prediction problem is oriented around—and evaluated in terms of—the identification of *pairs of nodes* that are likely to be connected. However, there is mounting evidence that the organization and evolution of networks is centered around higher-order interactions involving more than two nodes (Milo et al. 2002; Milo 2004; Benson et al. 2016, 2018; Lambiotte et al. 2019). In the case of social networks, triangles (cliques on three nodes) are extremely common due to various sociological mechanisms driving triadic closure (Easley and Kleinberg 2010; Holland and

NSF IIS-1422918, IIS-1546488, CCF-1909528, NSF Center for Science of Information STC, CCF-0939370, NASA, Sloan Foundation, DARPA SIMPLEX, NSF DMS-1830274, ARO W911NF-19-1-0057, and ARO MURI.

✉ Huda Nassar
hnassar@stanford.edu

Austin R. Benson
arb@cs.cornell.edu

David F. Gleich
dgleich@purdue.edu

[1] Stanford University, Stanford, USA

[2] Cornell University, Ithaca, USA

[3] Purdue University, West Lafayette, USA

Published online: 30 July 2020

Leinhardt 1977; Granovetter 1977; Rapoport 1953). Existing methods for link prediction are indeed motivated by these ideas. For instance, the Jaccard similarity between the sets of neighbors of two nodes—a common heuristic for link prediction (Liben-Nowell and Kleinberg 2007)—measures the number of triangles that would be created if the two nodes are linked, normalized by the total number of neighbors of the two nodes. Still, such methods are used to make predictions on *pairs of nodes*, rather than a prediction on the appearance of the higher-order structures directly. In our recent paper (Nassar et al. 2019), we develop a framework for directly predicting the appearance of a higher-order structure and we focus on the case of triangles. Here, we extend the methods we propose in Nassar et al. (2019) and our experimental setup in order to be able to predict triangles more reliably. In particular, the main differences from our previous paper (Nassar et al. 2019) are listed below.

- We modify our definition of neighborhood methods in this paper in order to capture predictions on edges that are not part of any triangles (our previous definitions were unable to make any predictions on such type of edges).
- We study the convergence and scalability of TRPR from Nassar et al. (2019), introduce an incremental update to it (we call the new method TRPRW), and find that TRPRW outperforms TRPR.
- We change our experimental setup to capture the prediction of adding two edges to the network, whereas in our previous experimental setup we allowed the prediction of either edges, or both.

Again, classical link prediction is centered around the following question: given a node u in the network, which nodes are likely to link to u? This scenario is illustrated in Fig. 1a. Our framing of the problem is similar, but we instead ask the following: given an edge (u, v) in the network, which nodes are likely to connect to both u and v? We call this the *pairwise link prediction problem*, and it is illustrated in Fig. 1b. There are several scenarios where the pairwise link prediction problem is natural, such as recommending a new friend to a couple on an online social network, recommending a movie to a couple in a video site, or predicting an effective drug given a disease-gene pair. We devise two new algorithms for the pairwise link prediction problem. The first is based on a variant of seeded (personalized) PageRank that uses multiple seeds, namely, one seed at each endpoint of the edge for which we are trying to predict new triadic connections. The second is based on a PageRank-like iteration that puts more weight on edges that participate in many triangles. In this sense, the method reinforces triangles, and we call the method "Triangle Reinforced PageRank" (TRPR). We compare these algorithms to natural extensions of local similarity measures that are common in link prediction, such as Jaccard similarity (Liben-Nowell and Kleinberg 2007), Adamic-Adar similarity (Adamic and Adar 2003), and preferential attachment (Newman 2001). For a given edge, each of the above methods produces a score for the remaining nodes in the graph. We find that our proposed diffusion-based methods are the least sensitive to the graph type and degree distribution and often produce the top results. We provide code for all the methods used in this paper in the repository: https://github.com/nassarhuda/pairseed

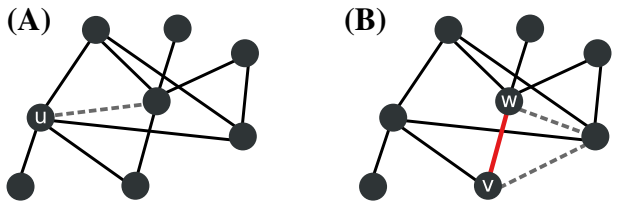

Fig. 1 **a** In standard link prediction, we are tasked with finding nodes that are likely to link to a given node u. **b** In this paper, we study pairwise link prediction, where we are tasked with finding nodes that are likely to form a triangle with a given edge (v, w)

2 Motivation

Since a network encodes pairwise relationships (edges) between elements (nodes), the link prediction problem is natural in many cases. Nevertheless, recent studies have shown that networks evolve through higher-order interactions, i.e., much of the structure in evolving networks involves interactions between more than just two nodes (Benson et al. 2018). Recent research has also introduced the problem of predicting the *time* when an edge addition will close a triangle (Dave and Hasan 2019). Furthermore, random graph models constructed from distributions of triangles have shown to be good fits for real-world data (Eikmeier et al. 2018), providing additional evidence that triadic relationships are important to the assembly of networks. These considerations motivate us to study higher-order generalizations of the link prediction problem.

3 Link prediction methods

We now briefly review some related work in link prediction. As part of this, we will go over methods that we will generalize in the next section for the pairwise link prediction problem. All of these methods assign some similarity score between pairs of nodes, where a larger similarity is indicative of pairs that are likely to connect. For notation, we use $\Gamma(u)$ to denote the set of neighbors of node u in the graph.

3.1 Local methods

Several approaches to link prediction are based on local information in the graph, namely a score is assigned to a pair of nodes w and u based on their neighborhoods $\Gamma(w)$ and $\Gamma(u)$.

One approach that falls under this category stems from the idea that as the set of nodes incident to both w and u, i.e., $|\Gamma(w) \cap \Gamma(u)|$, increases, the chance that w and u are connected also increases (Newman 2001). Here, $|\Gamma(w) \cap \Gamma(u)|$ is the number of triangles that would be formed if w and u were connected. Often, this number is normalized by the size of the neighborhoods, which gives rise to the Jaccard similarity between two nodes w and u: $\frac{|\Gamma(w) \cap \Gamma(u)|}{|\Gamma(w) \cup \Gamma(u)|}$.

The Adamic–Adar similarity measure (Adamic and Adar 2003) is another local score that assigns similarity scores between pairs of nodes based on the premise that when the common nodes between two nodes have little importance (where importance is measured by the logarithm of the degree), it is more likely that these two nodes are connected. The Adamic–Adar measure defines the similarity between two nodes w and u as $\sum_{z \in \Gamma(w) \cap \Gamma(u)} \frac{1}{\log(|\Gamma(z)|)}$. The logarithm function is used in an analogous way to how it is used in Information Retrieval, to dampen the importance of high degree nodes. As an example, the logarithm function will guarantee that the importance of a node with degree 1000 is close to the importance of a node with degree 1010, whereas two nodes with degree 10 and 20, will have more distant importance score. Even though in both scenarios the degree difference is the same, the importance difference is not.

There are two methods that are similar to the Adamic–Adar measure, that we discuss here. The first is the resource allocation index (Lü and Zhou 2011b), and its only difference from Adamic–Adar is that it removes the log function from the denominator of the Adamic–Adar equation. It can be generalized to perform the pairwise link prediction task in the same way we generalize the Adamic–Adar index in Sect. 4.1. The second metric is the Soundarajan Hopcroft index (Soundarajan and Hopcroft 2012). It uses the same concept from the resource allocation index but restricts predictions to happen within communities. This method, too, can be generalized to perform the pairwise link prediction task and can be used when community information is present.

A third local method is based on preferential attachment, where nodes are more likely to connect to *established* nodes in the network, and *established* nodes have a higher chance to connect to each other (Barabási and Albert 1999; Newman 2001). Using degree as a proxy for how established a node is, the preferential attachment score between nodes w and u is $|\Gamma(w)| \cdot |\Gamma(u)|$.

3.2 Global methods

Another set of approaches for link prediction are based on aggregating (weighted or normalized) path counts of varying lengths. In contrast to the local methods described above, these methods use global information about the entire network. For example, the Katz similarity counts the number of paths between two nodes, weighting paths of length k by β^k (Katz 1953; Liben-Nowell and Kleinberg 2007), where β is an attenuation parameter between 0 and 1. Another class of global methods are methods based on diffusions such as PageRank (Page et al. 1999). Such diffusion methods usually conserve a *seeded* amount of "mass" across the network, and for the task of link prediction, they are typically seeded by a particular node u. The similarity of u to all other nodes is then given by the amount of "mass" that diffuses to each other node. We will make use of PageRank-like methods in the next section.

4 Pairwise link prediction methods

We propose several methods for the pairwise link prediction problem. First, we extend the three local methods described above to measure node-edge similarity. After, we propose diffusion-based methods akin to seeded PageRank.

4.1 Local similarity measures for pairwise prediction

In Nassar et al. (2019), we extended the common local methods for link prediction to the pairwise link prediction paradigm. Through this extension, we needed to define the notion of a *neighborhood of an edge (u,v)*. Initially, the intersection of neighborhoods of the edge's endpoints was a natural choice, but in practice the intersection set is very limiting specially in scenarios when an edge is connected to the rest of the graph, yet does not participate in any triangles (for instance, Figs. 3 and 5 in Nassar et al. (2019)). So here, we revise this definition to capture the union of node neighborhoods rather than the intersection and state it below.

$$\begin{aligned}&\Gamma((u,v))\\&= \{z \mid z \text{ is connected to } u, v, \text{ or both, but is not } u \text{ or } v\}\\&= \Gamma(u) \cup \Gamma(v) \setminus \{u, v\}.\end{aligned}$$

Note that this is akin to the boundary of a set of vertices in the graph that is often used to define the size of a cut, which—for an edge—would correspond to the union of neighborhoods. Using the substitution gives us three similarity measures that will compute the similarity of an edge to a node.

– Jaccard Similarity (JS).

$$\mathrm{JS}(w,(u,v)) = \frac{|\Gamma(w) \cap \Gamma((u,v))|}{|\Gamma(w) \cup \Gamma((u,v))|}$$

– Adamic–Adar (AA).

$$\mathrm{AA}(w,(u,v)) = \sum_{z \in \Gamma(w) \cap \Gamma((u,v))} \frac{1}{\log|\Gamma(z)|}$$

– Preferential attachment (PA).

$$\mathrm{PA}(w,(u,v)) = |\Gamma(w)| \cdot |\Gamma((u,v))|$$

Further, we now introduce other variations of extending the common local methods for link predictions. Specifically, we extend the Jaccard similarity and Adamic–Adar measures to account for a combination of the single link prediction results. We use the maximum value of the single similarity score of both endpoints of an edge (u, v) with another node w, as well as the product of similarity values. We use these two functions, MAX and MUL, since they are non-linear combinations of the two single link prediction results. We state these measure below.

– Jaccard Similarity.

$$\mathrm{JS\text{–}MAX}(w,(u,v)) = \max(JS(w,u), JS(w,v))$$
$$\mathrm{JS\text{–}MUL}(w,(u,v)) = JS(w,u) \cdot JS(w,v)$$

– Adamic–Adar.

$$\mathrm{AA\text{–}MAX}(w,(u,v)) = \max(AA(w,u), AA(w,v))$$
$$\mathrm{AA\text{–}MUL}(w,(u,v)) = AA(w,u) \cdot AA(w,v)$$

Next, we discuss two methods for pairwise link prediction based on seeded PageRank, and use a combination of the single seeded PageRank results to compute a new measure of similarity between an edge and a node.

4.2 Pair-seeded PageRank

We now restate the pair-seeded PageRank method we use in Nassar et al. (2019). Seeded PageRank is a foundational concept in network analysis that models a flow of information in a network to predict links and communities on a network (Andersen et al. 2006; Gleich 2015). Seeded PageRank models information flow from the seed node to other nodes in the network via a Markov chain, and the stationary distribution of the chain provides the scores on the nodes. A high score on a node is a signal that the node should be connected to the seed node. More formally, let $\boldsymbol{A}$ be the symmetric adjacency matrix of an undirected graph, and let $\boldsymbol{P}$ be the column stochastic matrix of a random walk on that graph. Specifically, $P(i, j) = A(i, j)/|\Gamma(j)|$. Let u be the seed node. Then the seeded PageRank scores are entries of the solution vector $\mathbf{x}$ to the linear system

$$(\boldsymbol{I} - \alpha \boldsymbol{P})\mathbf{x} = (1-\alpha)\mathbf{e}_u.$$

Here, $\mathbf{e}_u$ is the vector of all zeros, except at index u, where $\mathbf{e}_u(u) = 1$ (i.e., $\mathbf{e}_u$ is the indicator vector on node u). The parameter α is the probability of transitioning according to the probability distribution in $\boldsymbol{P}$ and $(1 - \alpha)$ is the probability of teleporting according to the probability distribution in $\mathbf{e}_u$. The entries of $\mathbf{x}$ can be viewed as similarity scores between node u and the other nodes, in the same way $\mathbf{x}$ is used in Liben-Nowell and Kleinberg (2007), and thus these scores can be utilized for standard link prediction.

In the same way seeded PageRank predicts the relevance of other nodes in the network to a single seed node, we propose *pair-seeded PageRank* to predict the relevance of nodes to a single edge; with these similarities, we are able to make predictions for the pairwise link prediction problem. For a given edge (u, v), pair-seeded PageRank solves the following linear system:

$$(\boldsymbol{I} - \alpha \boldsymbol{P})\mathbf{x} = (1-\alpha)\mathbf{e}_{u,v}.$$

In this case, $\mathbf{e}_{u,v}$ is the vector of all zeros, except at indices u and v, where $\mathbf{e}_{u,v}(u) = \mathbf{e}_{u,v}(v) = 1/2$. The solution $\mathbf{x}$ can be interpreted as the similarity of each node to the edge (u, v). We note that pair-seeded PageRank is equivalent to the sum of single-seeded PageRank on each of the nodes, up to a scalar multiple.

Indeed, this is a useful and helpful observation as there are many systems designed to estimate large seeded PageRank values for single-seeds by using highly scalable random walk methods (Lofgren et al. 2016). Thus, this technique could be used wherever a PageRank-style prediction is already employed.

4.3 Extensions of single-seeded PageRank

We also use the single seeded PageRank solution of each endpoint of the edge we are interested in predicting links to and produce two more metrics for relating an edge to a node, in the same way we did with local methods. Denote $\mathbf{x}_u$, and $\mathbf{x}_v$ to be the seeded PageRank solutions for nodes u and v respectively. Then, we define *MAX* and *MUL* as follows.

$$\mathrm{MAX}(u,v) = \max(\mathbf{x}_u, \mathbf{x}_v) \text{ (element-wise maximum)}$$
$$\mathrm{MUL}(u,v) = \mathbf{x}_u \odot \mathbf{x}_v \text{ (element-wise multiplication)}$$

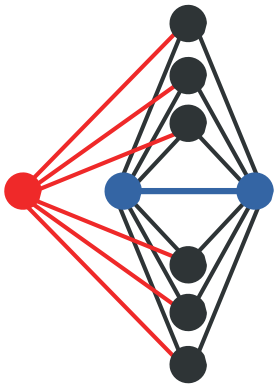

Fig. 2 Motivating social network example for the TRPR algorithm. If all of the *friends* of the blue couple know the red node, it is likely that the red node knows the blue couple as well. Running TRPR on the above example with $\mathbf{e}_{u,v}$ as the seed vector on the blue nodes reveals that the red node has the third highest score after the two blue nodes. After 10 iterations of Algorithm 1 with $\alpha = 0.85$, the output vector assigns a score of 0.120 to the red node, 0.062 to the black nodes, and 0.252 to the blue nodes

4.4 Triangle reinforced PageRank (TRPR)

We now revisit our method, triangle reinforced PageRank, TRPR from Nassar et al. (2019), extend the method to a weighted version, and introduce an empirical convergence study on this method. For an unweighted graph, the PageRank solution is highly affected by the degree of nodes in the network. Here, we *reinforce* the influence of triangles by giving edges participating in many triangles a higher weight. Figure 2 presents a motivating example for the usefulness of reinforcing triangles.

To develop our TRPR method, we first introduce a tensor $\underline{\boldsymbol{T}}$, that encodes all triangles in a network:

$$\underline{\boldsymbol{T}}(i,j,k) = \begin{cases} 1 & \text{if } (i,j,k) \text{ is a triangle} \\ 0 & \text{otherwise.} \end{cases}$$

Again, in our derivation, we assume that the graph is undirected so that $\underline{\boldsymbol{T}}$ is fully symmetric in all permutations of indices. A typical way to solve the PageRank linear system is the power method. With TRPR, we modify the power method by adding a step that redistributes the weights in the network. Specifically, we compute the matrix $\hat{\boldsymbol{X}} = \underline{\boldsymbol{T}}[\mathbf{x}]$, where $\hat{\boldsymbol{X}}(i,j) = \sum_k \underline{\boldsymbol{T}}(i,j,k)\mathbf{x}(k)$, which measures the relevance of edge (i,j) to the distribution of node scores in the vector $\mathbf{x}$. We then run an iteration of the power method on a weighted adjacency matrix $\boldsymbol{X} = \hat{\boldsymbol{X}} + \boldsymbol{A}$, where the columns are re-normalized to make the matrix column stochastic. Algorithm 1 shows the pseudocode.

A weighted version of TRPR. Although TRPR introduces higher weights to edges participating in many triangles by forming a new adjacency matrix $\hat{\boldsymbol{X}} + \boldsymbol{A}$, these weights are often dominated by the weights in the adjacency matrix $\boldsymbol{A}$. To give a fair contribution to these edges, we introduce a scalar multiple to $\hat{\boldsymbol{X}}$. Any scalar multiplied by $\hat{\boldsymbol{X}}$ that produces a fair contribution of weights from both, $\hat{\boldsymbol{X}}$ and $\boldsymbol{A}$, is what we are looking for and straightforward scalar we choose is $\gamma = \text{sum}(\boldsymbol{A})/\text{sum}(\hat{\boldsymbol{X}})$. This scalar will guarantee that the sum of weights in $\boldsymbol{A}$ and $\gamma\hat{\boldsymbol{X}}$ are equal. We present the pseudocode of the algorithm of the weighted version of TRPR in Algorithm 2.

Algorithm 1: TRPR

Input: $\underline{\boldsymbol{T}}$, adjacency matrix of undirected graph $\boldsymbol{A}$, α, $\mathbf{e}_{u,v}$, nb. iterations n
Output: $\mathbf{x}$
$\mathbf{x}_0 = \mathbf{e}_{u,v}$
for $i = 1, 2, \ldots, n$ **do**
  $\hat{\boldsymbol{X}}^{(i)} = \underline{\boldsymbol{T}}[\mathbf{x}_{i-1}]$ # i.e., $\hat{X}^{(i)}_{r,s} = \sum_k \underline{\boldsymbol{T}}(r,s,k)\mathbf{x}_{i-1}(k)$
  $\boldsymbol{P}_i = \text{normalize}(\hat{\boldsymbol{X}}^{(i)} + \boldsymbol{A})$ # column stochastic
  $\mathbf{x}_i = \alpha \boldsymbol{P}_i \mathbf{x}_{i-1} + (1-\alpha)\mathbf{x}_0$
return $\mathbf{x}_n$

Algorithm 2: TRPR-Weighted

Input: $\underline{\boldsymbol{T}}$, adjacency matrix of undirected graph $\boldsymbol{A}$, α, $\mathbf{e}_{u,v}$, nb. iterations n
Output: $\mathbf{x}$
$\mathbf{x}_0 = \mathbf{e}_{u,v}$
for $i = 1, 2, \ldots, n$ **do**
  $\hat{\boldsymbol{X}}^{(i)} = \underline{\boldsymbol{T}}[\mathbf{x}_{i-1}]$ # i.e., $\hat{X}^{(i)}_{r,s} = \sum_k \underline{\boldsymbol{T}}(r,s,k)\mathbf{x}_{i-1}(k)$
  $\gamma = \text{sum}(\boldsymbol{A})/\text{sum}(\hat{\boldsymbol{X}}^{(i)})$
  $\boldsymbol{P}_i = \text{normalize}(\gamma\hat{\boldsymbol{X}}^{(i)} + \boldsymbol{A})$ # column stochastic
  $\mathbf{x}_i = \alpha \boldsymbol{P}_i \mathbf{x}_{i-1} + (1-\alpha)\mathbf{x}_0$
return $\mathbf{x}_n$

TRPR and TRPRW can be implemented efficiently. Although TRPR involves the tensor $\underline{\boldsymbol{T}}$, we do not need to form it explicitly, and we show an alternative derivation here. We first unwrap one iteration of TRPR. Let $\boldsymbol{A}_i = \underline{\boldsymbol{T}}[\mathbf{x}_{i-1}] + \boldsymbol{A}$, at iteration i, and let $\boldsymbol{D}_i^{-1}$ be the diagonal matrix with the kth diagonal entry being the inverse of the sum of edge weights connected to node k in $\boldsymbol{A}_i$, we can translate $\mathbf{x}_i = \alpha \boldsymbol{P}_i \mathbf{x}_{i-1} + (1-\alpha)\mathbf{x}_0$ into

$$\mathbf{x}_i = \alpha((\underline{\boldsymbol{T}}[\mathbf{x}_{i-1}] + \boldsymbol{A})\boldsymbol{D}_i^{-1})\mathbf{x}_{i-1} + (1-\alpha)\mathbf{x}_0$$

Then,

$$\mathbf{x}_i = \alpha\underline{\boldsymbol{T}}[\mathbf{x}_{i-1}]\boldsymbol{D}_i^{-1}\mathbf{x}_{i-1} + \alpha\boldsymbol{A}\boldsymbol{D}_i^{-1}\mathbf{x}_{i-1} + (1-\alpha)\mathbf{x}_0.$$

Set $\mathbf{y}_{i-1} = \boldsymbol{D}_i^{-1}\mathbf{x}_{i-1}$. Then

$$\mathbf{x}_i = \alpha\underline{\boldsymbol{T}}[\mathbf{x}_{i-1}]\mathbf{y}_{i-1} + \alpha\boldsymbol{A}\mathbf{y}_{i-1} + (1-\alpha)\mathbf{x}_0.$$

The relevant computationally expensive pieces to compute are $\underline{\boldsymbol{T}}[\mathbf{x}_{i-1}]\mathbf{y}_{i-1}$ and the entries of $\boldsymbol{D}_i^{-1}$. Both involve the same type of operation. Using the definition of $\underline{\boldsymbol{T}}[\mathbf{x}]$ we have that the matrix-vector product $\mathbf{z} = \underline{\boldsymbol{T}}[\mathbf{x}]\mathbf{y}$ has $z_i = \sum_j \sum_k \underline{\boldsymbol{T}}(i,j,k)y(j)x(k)$. Consequently, if we have any means of *iterating* over the triangles of a graph, then we can

compute $\underline{\boldsymbol{T}}[\mathbf{x}]\mathbf{y}$ for any pair $\mathbf{x}$ and $\mathbf{y}$ in a fashion akin to a sparse-matrix-vector product but in runtime proportional to the number of triangles in the graph.

This directly enables us to compute $\underline{\boldsymbol{T}}[\mathbf{x}_{i-1}]\mathbf{y}_{i-1}$. To compute the entries in $\boldsymbol{D}_i^{-1}$, note that $\underline{\boldsymbol{T}}[\mathbf{x}]$ is a symmetric matrix because it can be written as a sum of symmetric matrices (since $\underline{\boldsymbol{T}}$ is fully symmetric in all permutations). Thus, the row-sums of $\boldsymbol{A}_i$ are the vertex-degrees we need to build $\boldsymbol{D}_i^{-1}$. Let $\mathbf{e}$ be the vector of all ones; these row sums are computed as $\boldsymbol{A}_i\mathbf{e} = \underline{\boldsymbol{T}}[\mathbf{x}_i]\mathbf{e} + \boldsymbol{A}\mathbf{e}$. Since $\boldsymbol{A}$ is not changing, we only need to compute the column sums of $\underline{\boldsymbol{T}}[\mathbf{x}_i]\mathbf{e}$ at each iteration. Again, we can use an implicit tensor-vector-vector product operation to compute the column sums. And thus, all operations involving the tensor $\underline{\boldsymbol{T}}$ are linear in terms of the number of triangles in the network, and we use a fast routine to iterate through triangles in a graph. For ease of reuse, we provide the code for TRPR[1].

TRPR runs in time proportional to the number of triangles. We experimentally validate the running time of TRPR on generalized preferential attachment graphs (Avin et al. 2015) while varying the size of the graphs. In this experiment, we vary the edge addition probability p_e, and allow the node addition probability to be $1 - p_e$. We use p_e values = $\{0.5, 0.6, 0.7, 0.8, 0.9\}$ and graph sizes $n = \{2000, 5000, 10,000, 20,000, 50,000, 100,000\}$ and for every pair of (n, p_e), we count the number of triangles produced. In Figure 3, we show the running time in seconds (y-axis) as the number of triangles increases (with the increase of the edge addition probability) and it shows that TRPR scales linearly with the increase of the number of triangles and is a fast method when implemented efficiently.

We run TRPR with 10 iterations. While we still seek a robust convergence theory for the TRPR iteration, at the moment, we run this algorithm for 10 iterations. This choice was based on the following experiment, where we study the ordering of nodes from every iteration. In this experiment, we study the ordering of the nodes from every iteration and notice that the order does not change much after just a few iterations. In Fig. 4, we show the Spearman's rank correlation coefficient and the Kendall rank correlation coefficient between two consecutive iterates from TRPR on 4 graphs from the experiments section. We notice that after a few iterations, 10 iterations, the orderings of the vectors no longer change (red solid line in Figure 4).

Convergence of TRPR. Convergence of this type of nonlinear system of equations is theoretically delicate with bounds that are often insufficient for practice (Benson et al. 2017). As stated, with both starting point and number of iterations fixed, TRPR produces a unique deterministic and reproducible set of scores that locally capture the influence of both the graph and the reinforced triangles. As the number of iterations grows, empirically, we observe that the iterations converge. See our evidence in Fig. 5, where we show the 1-norm difference (sum of the absolute value of the difference vector between two consecutive iterates) decay from two consecutive iterates from TRPR on 4 datasets used in the experiments section.

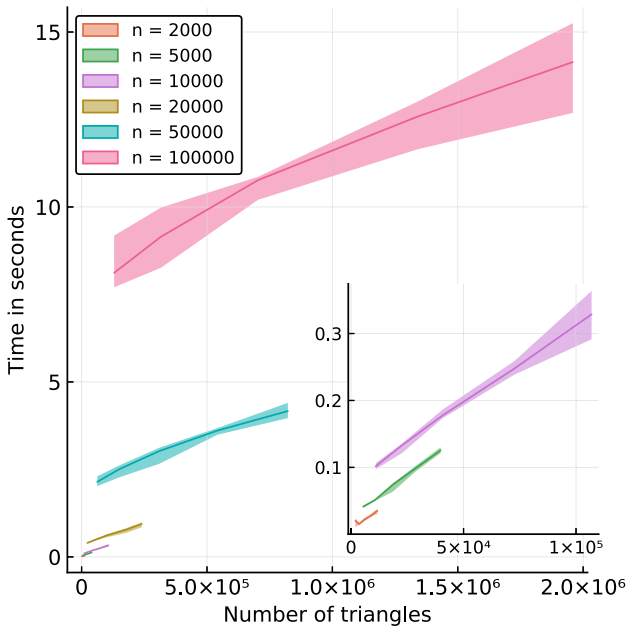

Fig. 3 Time in seconds for running TRPR with 10 iterations on generalized preferential attachment graphs of different sizes as the number of triangles increases (due to increasing the edge addition probability in the generalized preferential attachment model). The various colors correspond to different graph sizes, as shown in the legend. And the inset figure is a zoom in on the lower left corner of the plot. The shaded area is 20th and 80th percentiles while the solid line is the median time after running the same experiment for 20 trials. This figure shows that TRPR scales linearly with the increase of the number of triangles and is a fast method which can scale to large graphs

5 Experimental setup

We now perform a series of experiments on synthetic as well as real-world graphs from a variety of disciplines, including online social networks, communication networks, and biological interaction networks. We also include experiments for static networks as well as a temporal network. The main difference in our experimental setup here from our previous paper (Nassar et al. 2019), is the focus on predicting both edges connected to the endpoints of a given edge and we discuss the details later in this section. Throughout this section, we use the term *seed edge* to signify that this is the edge we would like to make predictions on. When we refer to node predictions given an edge (u, v), this is analogous to predicting two edges, one from the predicted node to u and another from the predicted node to v. For a given edge (u, v), we use

[1] https://github.com/nassarhuda/pairseed/blob/master/trpr.jl

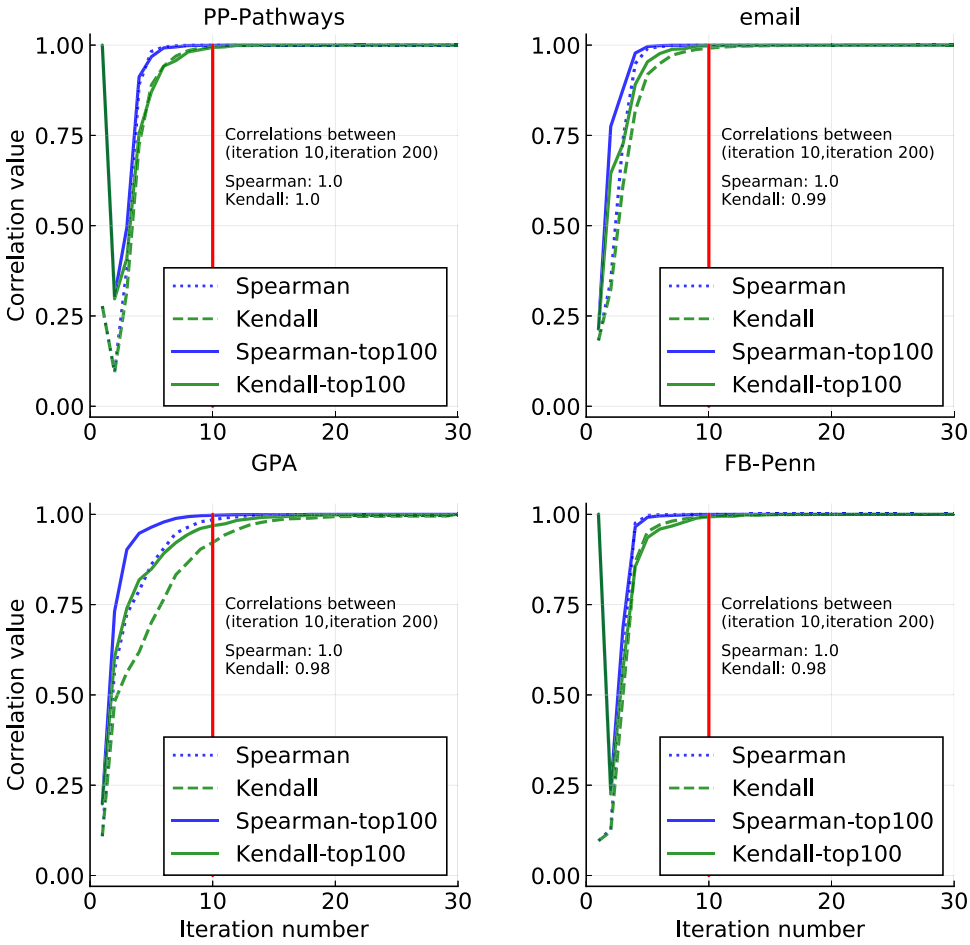

Fig. 4 Spearman's rank correlation coefficient and the Kendall rank correlation coefficient between consecutive iterates from TRPR. The solid plots show the consecutive correlation values when truncating the vectors to take the top 100 nodes, and the dashed lines compare the orderings in the full vectors. The vertical red line represents the 10th iterate. The text in the figures is the correlation between the 10th iterate and the 200th iterate. These correlations support our choice of 10 iterations in the experiments involving TRPR

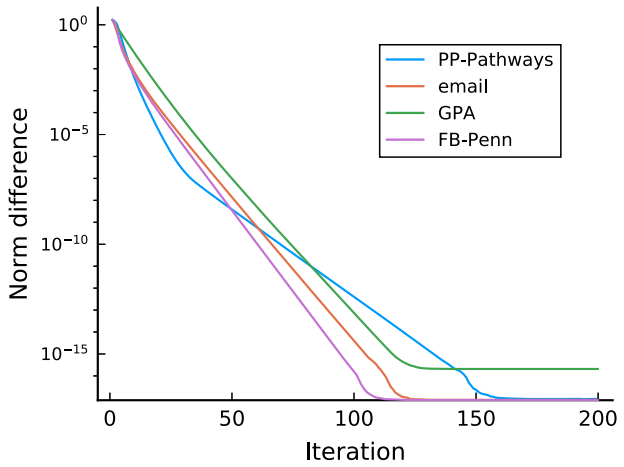

Fig. 5 1-norm convergence of TRPR on 4 datasets of varying sizes used in the experiments section. These plots show that TRPR reaches a steady state experimentally

the Success Probability (SP) measure, which we define as follows:

$$\mathrm{SP}((u,v),k) = \begin{cases} 1 & \text{if at least one ground truth node } w \text{ appears in the top } k \\ & \text{predictions for edge } (u,v) \\ 0 & \text{otherwise.} \end{cases}$$

For our experiments, we want to assess the efficacy of the methods presented so far, and a common way to achieve this in link prediction problems is by splitting the set of edges of a graph into two sets (e.g., Liben-Nowell and Kleinberg 2007). The input graph to our algorithms will be comprised of the edges in one set (the bigger set), and the remaining edges (the smaller set) will be treated as *missing edges*. Then, for a given edge in the input graph, the goal is to predict nodes that connect to both endpoints of this edge. To inspect whether the predicted edges were in fact the relevant edges to predict, we look in the missing edges set to see if our predictions are present there.

Each method presented so far gives an ordering of the nodes based on the similarity scores computed. Our prediction set will be the top k nodes with the highest similarity scores, and that aren't already connected to either endpoint of the seed edge in the input graph. A correct prediction here would be a node which forms a triangle with the seed edge using two edges from the missing edges set. For each graph, we run 500 random experiments and we try to predict links to 500 randomly chosen seed edges. For each of these experiments, an SP value (0 or 1) is computed, based on whether we found at least one correct prediction (SP value of 1) or

not (SP value of 0). The overall score is then computed as the mean value over all experiments.

The main choice for this measure in contrast to the area under ROC curve (AUC score) measure we use in Nassar et al. (2019), is the small number of nodes that we often want to recover. For a given edge (u, v) in the input graph, the missing edges set must have the edges (u, w) and (v, w) for w to be considered a correct ground truth node for recovery. In subsequent sections, we will see that the number of such candidate nodes in the set of missing edges is often small (1 in most instances), and thus a measure such as the AUC score does not fully capture the performance of our methods.

5.1 Leave One Edge's Triangles Out (LOETO)

The LOETO experiments are akin to the leave-p-out cross validation method, in the sense that we will leave p edges out and treat them as *missing edges*, and the remaining edges will constitute the input graph. Here, $p = (2 \times$ number of nodes that form a triangle with a randomly chosen edge). An experimental trial in this setting is designed as follows. Randomly pick a seed edge in the graph, and find all the wedges (path of length 2) that form a triangle with this edge. Next, remove all these wedges and treat them as missing edges. Figure 6 visualizes this experiment. The graph used will be the one in panel B of Fig. 6 (the grey dashed edges no longer appear in the network and the goal is to recover the connections with the green nodes in the graph). We then use the pairwise link prediction methods on the seed edge, which produces an ordering on the nodes, and given this ordering, we compute the success probability. Since this method leaves a big portion of the graph in the input graph, we compute its success probability with top $k = 5$.

5.2 Hold-out cross validation

In this setup, for a given network, we remove 30% of the edges and treat them as missing edges, and use the remaining 70% of the edges to constitute the input graph. Note that we increase the split for our missing edges set from Nassar et al. (2019) since here, we are interested in finding nodes that connect to both endpoints of a given edge. And increasing the size of the missing edges set increases the possible number of nodes that fit this criteria for our algorithms to find. An experimental trial in this setting is designed as follows. For a random seed edge in the input graph, we use the pairwise link prediction methods to predict which nodes will form triangles with the randomly chosen seed edge. Each method produces a similarity score on all nodes, and we use the ordering of the nodes induced by the scores to calculate the Success Probability with top k values $= 5, 25$.

We also perform a similar experiment on temporal networks with timestamps on the edge arrivals. In this scenario, we split the data based on these arrival timestamps—the first 70% of the edges to appear in time constitute the input graph to our methods, and the later 30% are treated as missing edges. In this set of experiments, we perform one more processing step to guarantee that the network we will use as input graph is connected. If the network is disconnected, we extract the largest connected component.

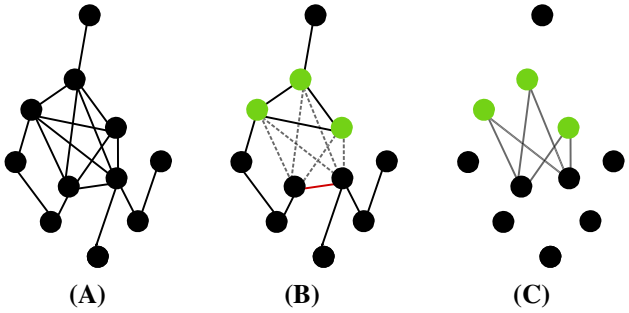

Fig. 6 Illustration of the Leave One Edge's Triangles Out (LOETO) experiment. For a given graph (subfigure **a**), randomly pick an edge (red edge in subfigure **b**) and remove all edges that form a triangle with it (dashed gray lines in subfigure **b**). Run all our methods on this new graph. The edges to predict are the ones shown in subfigure **c**

5.3 Summary of methods and parameter settings

Finally, we summarize all of the methods that we use for pairwise link prediction.

- *Pairseed* This is our method described in Sect. 4.2. We use the implementation from `MatrixNetworks.jl` (Nassar and Gleich 2018) with $\alpha = 0.85$. This implementation solves the linear system until convergence to machine precision.
- *TRPR* This is our method described in Sect. 4.4. We use $\alpha = 0.85$ and number of iterations $n = 10$.
- *TRPRW* This is the modified weighted version of the TRPR algorithm described in Sect. 4.4 as well. We use $\alpha = 0.85$ and number of iterations $n = 10$.
- *MUL, MAX* These are the methods from Sect. 4.3 that extend the single-seeded PageRank solutions. We use the same implementation used by Pairseed, with $\alpha = 0.85$.
- *AA, PA, JS* For a seed edge, we compute the generalized Adamic–Adar, preferential attachment, and Jaccard similarity scores, respectively (as presented in Sect. 4.1) between the seed edge and all remaining nodes in the graph.
- *AA–MUL, AA–MAX, JS–MUL, JS–MAX* These are the methods from Sect. 4.1, and they use the single node similarity from both endpoints of a seed edge to compute a new measure of similarity.

Table 1 Statistics of the real-world datasets used in this paper

Network name	Nodes	Edges	Triangles	Type
Penn94	41536	1362220	7207796	Social
Caltech36	762	16651	119562	Social
Ch-Ch-Miner	1510	48512	568466	Biology
P-P-Pathways	21521	338624	2394642	Biology
email	1133	5451	5343	Communication
CollegeMsg	1899	13838	14319	Temporal
Email-EU	1005	32128	105461	Temporal

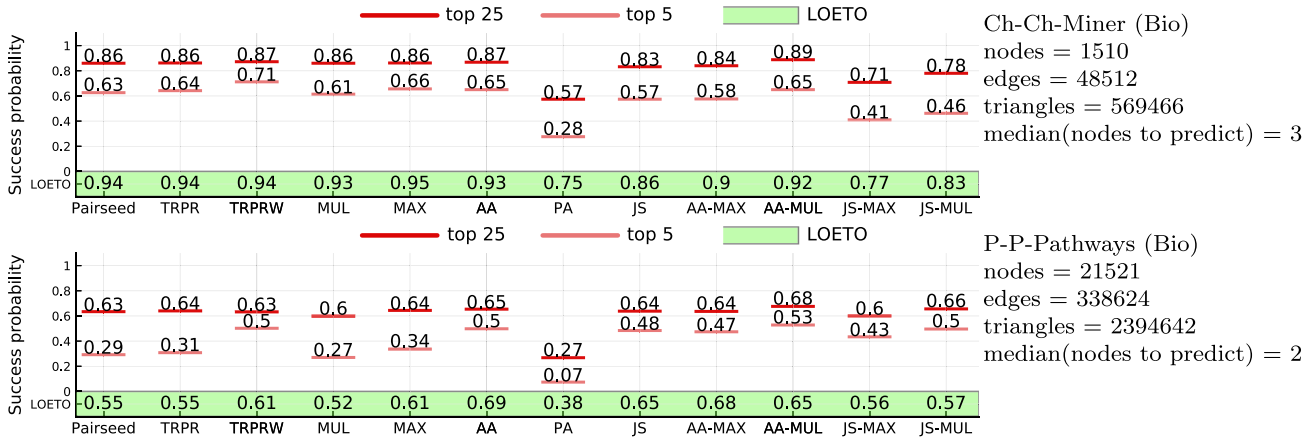

Fig. 7 Success probability results for the two biological datasets. In both datasets, we notice that TRPRW outperforms the remaining diffusion type methods and performs best on the top k predictions metric on the Ch-Ch-Miner dataset. Another method that stands out in these two datasets is AA-MUL which is the best method in terms of top k predictions in the P-P-Pathways dataset, with TRPRW performing worse than AA-MUL by around 5% on the top k measures

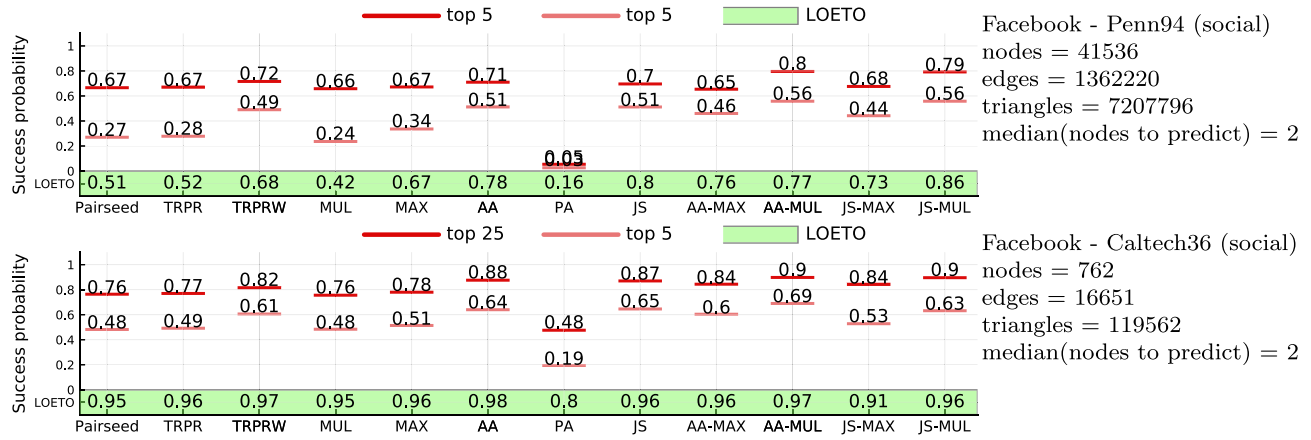

Fig. 8 Success probability results for the two social networks datasets. In both datasets, we notice that local methods generally outperform diffusion type methods. This is mainly due to how social networks grow (Schoenebeck 2013) and the influence of neighbors of nodes for making new connections. Here too, TRPRW outperforms other diffusion type methods and produces comparable results to the best local methods on the top 25 and LOETO measures

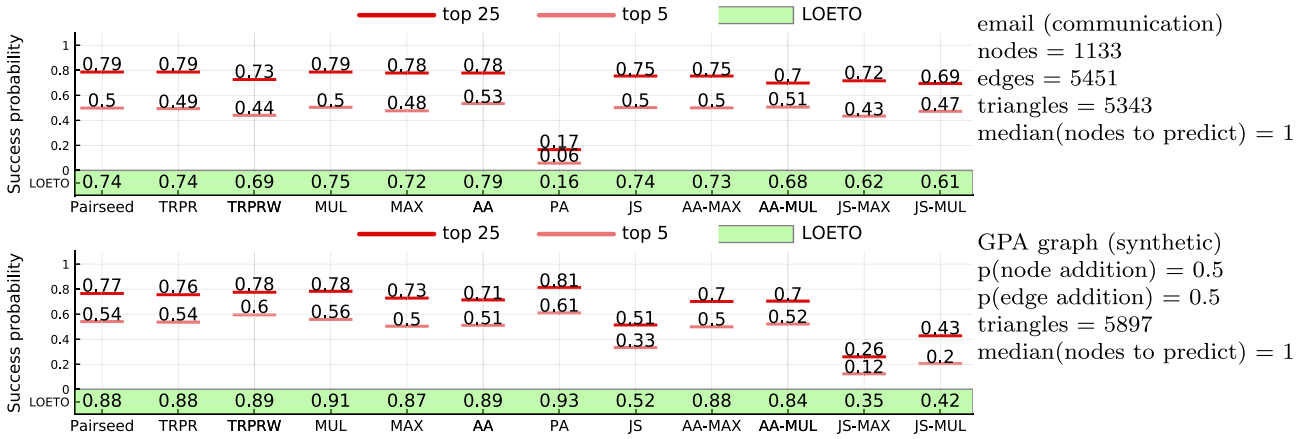

Fig. 9 Success probability results for the two networks, email and an instance of a GPA graph. We group these two graphs together because they have a very small number of triangles compared to the other networks. In these datasets TRPRW does not contribute an improvement over the other diffusion type methods. In the email network, TRPR performs best in the top k metric, and TRPRW performs best after the PA method on the GPA graph

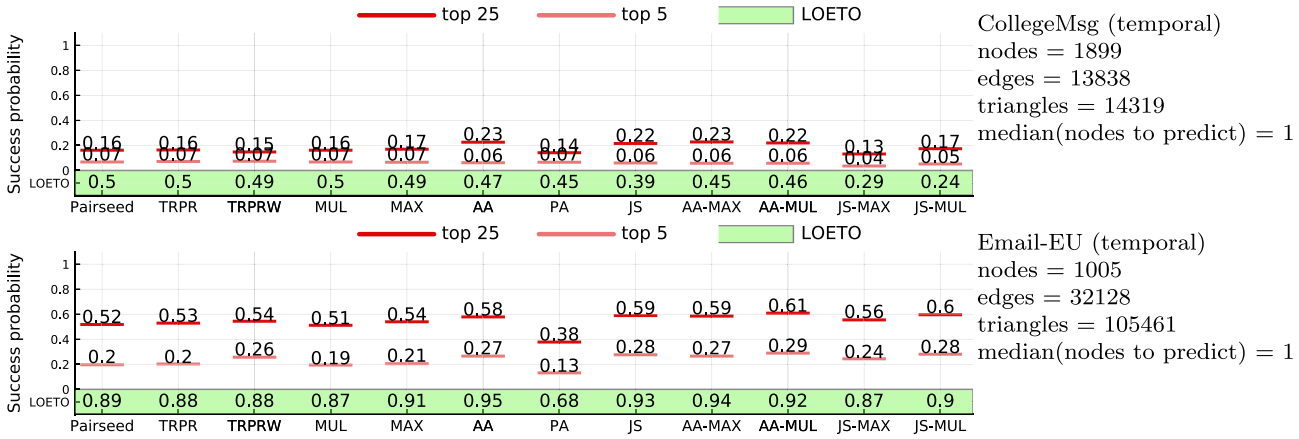

Fig. 10 Binary Mean Value results for the two temporal networks, CollegeMsg and Email-EU. The results on temporal networks are generally worse than the results on static networks, and this can be an indicator that our methods are stronger in predicting missing links rather than future links

6 Pairwise link prediction results

In all of the results in this section, we report the success probability from our predictions over 500 random experiments. We use seven real-world graphs from different disciplines in this section and give a summary of their statistics in Table 1. We also use a synthetic graph generated from the generalized preferential attachment model (GPA) (Avin et al. 2015).

Synthetic graph. generalized preferential attachment (GPA) (Newman 2001) is a synthetic graph generation model that generalizes the classical preferential attachment model to allow for the addition of new components at each step of the algorithm. For our experiments, we generate a graph with 5000 nodes and allow the event of node addition with probability 1/2, and we allow the event of edge addition with probability 1/2. The starting graph structure is a clique of size 5. At each step of the graph generation process, an edge or node is added by attaching proportionally to the degrees of the existing nodes.

Real world graphs. We use various real world graphs to test our methods and provide statistics about them in Table 1. *Penn94* and *Caltech36* are online social networks from the *Facebook100* collection of datasets (Traud et al. 2012). These two datasets are the biggest and smallest networks in terms of number of nodes respectively from this collection. *Ch-Ch-Miner* is a biological network of drug (chemical) interactions (Wishart et al. 2017; Stanford SNAP Group 2017). *P-P-Pathways* is a biological network of physical interactions between proteins in humans (Agrawal et al. 2018). *email* is an email communication network (Guimerà et al. 2003). Finally, *CollegeMsg* (Panzarasa et al. 2009) and *email-EU* (Panzarasa et al. 2009) are temporal net-

works representing private messages (CollegeMsg) or emails (email-EU) between users in a network.

Results. We show the results of all methods in Figs. 7, 8, 9, and 10. Overall, we notice that the diffusion methods have more consistency in performance compared to local measures. For instance, AA-MUL — which is one of the best performers on some datasets (P-P-Pathways in top 25 and top 5 metrics) — drops to be one of the worst performers in the top 25 metric on the email dataset. PA is the best performer on the GPA model, but drops to be the worst performer on all other graphs. In contrast, TRPRW performs best on the Ch-Ch-Miner and email datasets but never drops to be one of the worst methods on any of the datasets. Temporal graphs (CollegeMsg and Email-EU) both suffered from lower top k scores as compared to static graphs, which suggests that our methods are possibly stronger in detecting missing links rather than future links. Upon further investigation on the temporal graphs, we found that most of the top k predictions were at least two hops away from the seed edges. In the temporal data, these wedges (length-2 paths) did not close to form triangles and thus the prediction was incorrect according to the timestamped data. TRPRW seemed to improve the performance of TRPR in general but did not contribute an improvement on the email and GPA networks. Upon looking closely at these two networks, we found that the number of triangles is very small and thus using the unweighted TRPR version which is close in performance to Pairseed, is more ideal on datasets that do not contain many triangles.

To summarize, the diffusion type methods (Pairseed, TRPR, and TRPRW) seem to have the most consistent results across all experiments, with TRPRW performing better when the number of triangles in the network is large. In contrast, local methods (such as ones related to Adamic–Adar and the Jaccard similarity) seem to be reliable for predictions on social networks specifically and this mainly related to how social networks grow (Schoenebeck 2013).

7 Back to standard link prediction

In this section we bring our attention back to the standard link prediction problem and show how the methods we presented in this paper can also be used to further enhance standard link prediction. We split our data in the same way to the previous experiments except that here we use an 80–20 split. This is because we no longer need to find paths of length 2 for prediction purposes and we can thus increase the set of missing edges by 10%. Then, for the top 100 nodes with the largest degree in the input graph, we perform different types of seeded PageRank diffusion for link prediction on these nodes. This choice of nodes serves the purpose of identifying nodes that have a higher chance of making connections in the test data. We measure performance in terms of Area Under the ROC curve (henceforth, AUC score). Our baseline is single-seeded PageRank.

Table 2 Description of methods inspired by pairwise link to perform the standard link prediction task

sum▲	For a certain node i, aggregate the pair-seeded PageRank results from all edges adjacent to i. This is equivalent to performing PageRank with a normalized initial vector valued 1 at the indices of all the neighbors of i, and degree(i) at index i.
max●	This is similar to the previous approach, but here, we instead take the element-wise maximum value of the pair-seeded PageRank vectors.
star-seed+	This is similar to pair-seeded PageRank, except that we start PageRank with a normalized initial vector valued 1 at the index of the seed node and all its neighbors.
TRPR♦	This uses the same starting vector used by star-seed, but instead, applies the TRPR algorithm on it.

Our results on pairwise link prediction suggest that multiple seeds with PageRank-like methods are effective for prediction. Here, we consider four different multiple-seeding strategies and compare them to single-seeded PageRank for the classical link prediction problem. We summarize the four new methods in Table 2. The methods *sum*, *max*, and *star-seed* are motivated by the double seeding idea used in the previous sections.

We use real-world networks from Sect. 6, and present our results in Fig. 11. The scatter plots compare the AUC score of the neighborhood-based seeding methods to the AUC scores from single-seeded PageRank. These results suggest that neighborhood-based seeding is superior to single-seeded PageRank as a link prediction method.

8 Discussion and future work

Link prediction is a well-studied research topic due to its utility in many disciplines. Traditional link prediction methods aim to find pairs of nodes that are likely to form a link. Here, we studied a higher-order version of the problem called pairwise link prediction where we predict nodes that are likely to form a triangle with an edge.

In our previous work, (Nassar et al. 2019), we generalized local link-prediction methods and we developed two PageRank-based methods for this problem. In this

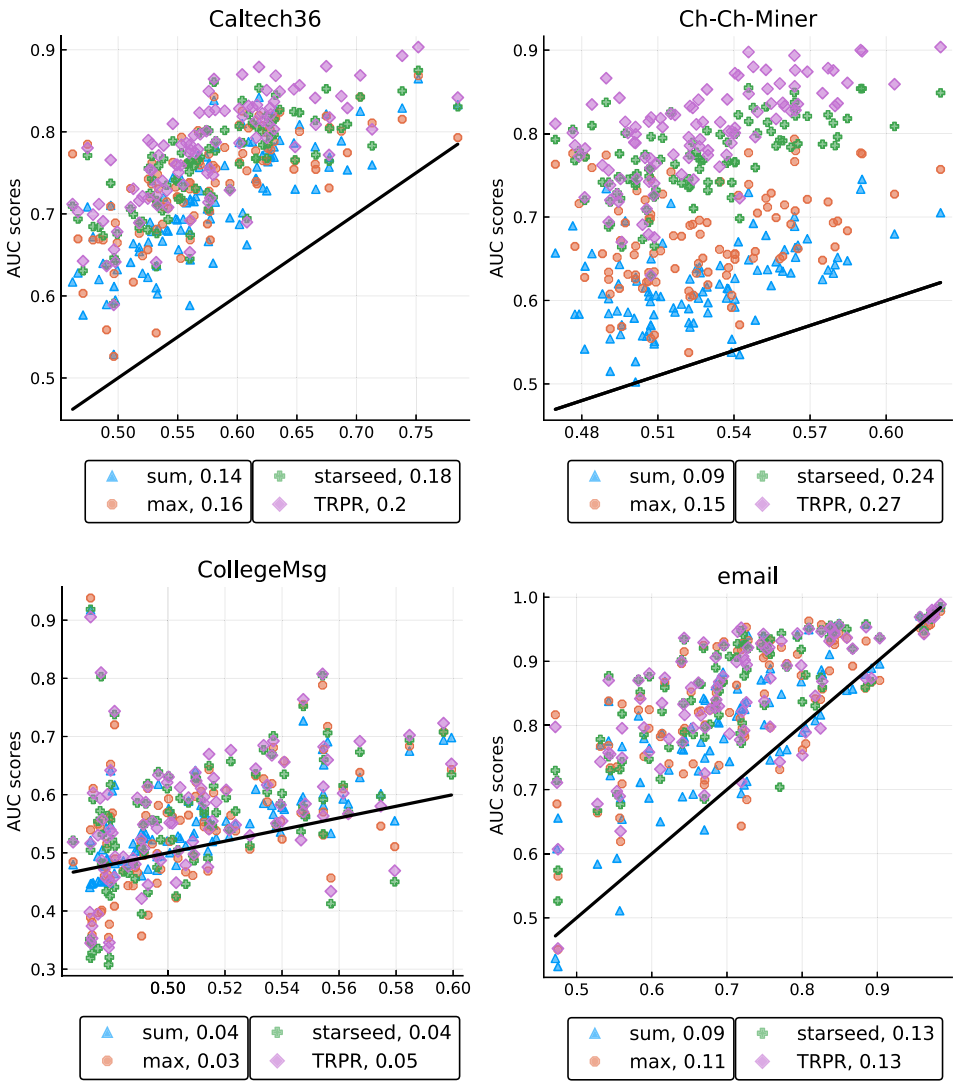

Fig. 11 Results of standard link prediction experiment on four real-world networks. Each scatter plot shows the link prediction AUC results of 100 experiments of methods inspired by our pairwise link prediction proposal with respect to the AUC scores of single-seeded PageRank. The solid black line is the plot of $f(x) = x$. Points above the line are cases where our proposed methods have superior performance to standard single-seeded PageRank. We see that in most cases the four methods outperform the classical seeded PageRank method. This study suggests that it is useful to consider a node's neighborhood for the purposes of seeding for link prediction with PageRank. The values in the legend serve as a summary performance measure, which is the average distance to the $f(x) = x$ line

manuscript, we revised our generalizations of the local link-prediction methods and added new extensions to these methods. Further, we included a more detailed analysis of the convergence and scalability of TRPR from Nassar et al. (2019), and introduced an incremental update to it, namely TRPRW. In general, the PageRank-based methods remained consistent in behavior on a variety of datasets. Using these results as inspiration, we then developed multiple-seeding strategies for PageRank in classical link prediction, which outperform their standard single-seeded counterparts.

TRPR (Triangle Reinforced PageRank) is our new principled method for the task of pairwise link prediction. We demonstrated that TRPR is computationally efficient; the implementation details of TRPR can improve on the idealized algorithm by taking advantage of a triangle iterator that avoids building a tensor. We also presented a weighted version of TRPR (TRPRW) which gave equal weighting contributions from edges and triangles. We note that highly efficient implementations of our procedures are possible given their close relationships with traditional PageRank methods. Scaling to billions of nodes and edges is simply not a problem given current abilities to compute PageRank (e.g., Lofgren et al. 2016), and especially that we have an existing routine to iterate through triangles in a graph quickly. Even though we do not provide theoretical evidence on the convergence of TRPR, we presented a discussion that shows that, in fact, TRPR appears to converge and reach a steady state empirically.

In this paper, we focused on the problem of predicting a node to connect to the endpoints of a given edge, and an alternate problem is to predict an edge that is important when given a single node. We intend to extend this work to the latter scenario by adapting TRPR to this purpose. TRPR can be adapted by fixing one seed node and using the values in the matrix $\hat{X}^{(i)}$ from Algorithms 1 and 2 as similarity measures between edges and the seed node. Finally, the work we present in this paper provides a framework for higher-

order link prediction that is not limited to triangles, and the space of higher-order prediction problems has limitless substructure.

References

Adamic LA, Adar E (2003) Friends and neighbors on the web. Soc Netw 25(3):211–230

Agrawal M, Zitnik M, Leskovec J (2018) Large-scale analysis of disease pathways in the human interactome. In: Pacific symposium on biocomputing, vol 23, p 111. World Scientific

Andersen R, Chung F, Lang K (2006) Local graph partitioning using PageRank vectors. In: 2006 47th annual IEEE symposium on foundations of computer science. IEEE

Avin C et al (2015) Core size and densification in preferential attachment networks. In: International colloquium on automata. languages, and programming, ICALP 2015. Springer, Berlin, pp 492–503

Backstrom L, Leskovec J (2011) Supervised random walks: predicting and recommending links in social networks. WSDM '11, pp 635–644. ACM, New York, NY, USA

Barabási A.L, Albert R (1999) Emergence of scaling in random networks. Science 286(5439), 509–512 . https://science.sciencemag.org/content/286/5439/509.full.pdf

Benson AR, Abebe R, Schaub MT, Jadbabaie A, Kleinberg J (2018) Simplicial closure and higher-order link prediction. Proc Natl Acad Sci 115:E11221–E11230

Benson AR, Gleich DF, Leskovec J (2016) Higher-order organization of complex networks. Science 353(6295):163–166

Benson AR, Gleich DF, Lim LH (2017) The spacey random walk: a stochastic process for higher-order data. SIAM Rev 59(2):321–345

Clauset A, Moore C, Newman MEJ (2008) Hierarchical structure and the prediction of missing links in networks. Nature 453:98. https://doi.org/10.1038/nature06830

Dave V, Hasan M (2019) Triangle completion time prediction using time-conserving embedding. ECMLPKDD

Easley D, Kleinberg J (2010) Networks, crowds, and markets: reasoning about a highly connected world. Cambridge University Press, Cambridge

Eikmeier N, Ramani AS, Gleich DF (2018) The hyperkron graph model for higher-order features. In: IEEE international conference on data mining, ICDM 2018, Singapore, 2018

Gleich DF (2015) PageRank beyond the web. SIAM Rev 57(3):321–363

Gomez-Uribe CA, Hunt N (2015) The netflix recommender system: algorithms, business value, and innovation. ACM Trans Manag Inf Syst 6(4):13:1–13:19

Granovetter M.S (1977) The strength of weak ties. In: Social networks, pp 347–367. Elsevier

Guimerà R, Danon L, Díaz-Guilera A, Giralt F, Arenas A (2003) Self-similar community structure in a network of human interactions. Phys Rev E 68:065103

Holland P.W, Leinhardt S (1977) A method for detecting structure in sociometric data. In: Social networks, pp 411–432. Elsevier

Katz L (1953) A new status index derived from sociometric analysis. Psychometrika 18(1):39–43

Lambiotte R, Rosvall M, Scholtes I (2019) From networks to optimal higher-order models of complex systems. Nat Phys 15(4):313–320

Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inf Sci Technol 58(7):1019–1031

Lin C.H, Konecki D.M, Liu M, Wilson S.J, Nassar H, Wilkins A.D, Gleich D.F, Lichtarge O (2018) Multimodal network diffusion predicts future disease-gene-chemical associations

Lofgren P, Banerjee S, Goel A (2016) Personalized pagerank estimation and search: A bidirectional approach. In: Proceedings of the ninth ACM international conference on web search and data mining, WSDM '16, pp 163–172. ACM, New York

Lü L, Zhou T (2011) Link prediction in complex networks: a survey. Phys A Stat Mech Appl 390(6):1150–1170

Lü L, Zhou T (2011b) Link prediction in complex networks: a survey. Phys A 390(6):1150–1170

Milo R (2004) Superfamilies of evolved and designed networks. Science 303(5663):1538–1542

Milo R, Shen-Orr S, Itzkovitz S, Kashtan N, Chklovskii D, Alon U (2002) Network motifs: Simple building blocks of complex networks. Science 298(5594), 824–827. https://science.sciencemag.org/content/298/5594/824.full.pdf

Nassar H, Benson AR, Gleich DF (2019) Pairwise link prediction. ASONAM

Nassar H, Gleich DF (2018) Matrixnetworks.jl. https://github.com/nassarhuda/MatrixNetworks.jl

Newman MEJ (2001) Clustering and preferential attachment in growing networks. Phys Rev E 64:025102

Page L, Brin S, Motwani R, Winograd T (1999) The pagerank citation ranking: bringing order to the web. Technical report 1999-66, Stanford InfoLab . Previous number = SIDL-WP-1999-0120

Panzarasa P, Opsahl T, Carley KM (2009) Patterns and dynamics of users' behavior and interaction: network analysis of an online community. J Am Soc Inform Sci Technol 60(5):911–932

Rapoport A (1953) Spread of information through a population with socio-structural bias: I. assumption of transitivity. Bull Math Biophys 15(4):523–533

Schoenebeck G (2013) Potential networks, contagious communities, and understanding social network structure. In: Proceedings of the 22nd international conference on World Wide Web, WWW '13, pp 1123–1132. Association for computing machinery, New York. https://doi.org/10.1145/2488388.2488486

Soundarajan S, Hopcroft J (2012) Using community information to improve the precision of link prediction methods. In: Proceedings of the 21st international conference on World Wide Web, pp 607–608

Stanford SNAP Group: (2017) Miner: Gigascale multimodal biological network. https://github.com/snap-stanford/miner-data

Traud AL, Mucha PJ, Porter MA (2012) Social structure of facebook networks. Phys A Stat Mechan Appl 391(16):4165–4180. https://doi.org/10.1016/j.physa.2011.12.021. http://www.sciencedirect.com/science/article/pii/S0378437111009186

Wishart DS et al (2017) DrugBank 5.0: a major update to the DrugBank database for 2018. Nucleic Acids Res 46:D1074–D1082