Noname manuscript No.
(will be inserted by the editor)

Supervised and Extended Restart in Random Walks for Ranking and Link Prediction in Networks

Woojeong Jin · Jinhong Jung · U Kang

Received: date / Accepted: date

Abstract Given a real-world graph, how can we measure relevance scores for ranking and link prediction? Random walk with restart (RWR) provides an excellent measure for this and has been applied to various applications such as friend recommendation, community detection, anomaly detection, etc. However, RWR suffers from two problems: 1) using the same restart probability for all the nodes limits the expressiveness of random walk, and 2) the restart probability needs to be manually chosen for each application without theoretical justification.

We have two main contributions in this paper. First, we propose Random Walk with Extended Restart (RWER), a random walk based measure which improves the expressiveness of random walks by using a distinct restart probability for each node. The improved expressiveness leads to superior accuracy for ranking and link prediction. Second, we propose SuRe (Supervised Restart for RWER), an algorithm for learning the restart probabilities of RWER from a given graph. SuRe eliminates the need to heuristically and manually select the restart parameter for RWER. Extensive experiments show that our proposed method provides the best performance for ranking and link prediction tasks, improving the MAP (Mean Average Precision) by up to 14.7% on the best competitor.

Keywords Random walk with restart · Ranking · Link prediction

1 Introduction

How can we measure effective node-to-node proximities for graph mining applications such as ranking and link prediction? Measuring relevance (i.e., proximity or similarity) scores between nodes is a fundamental tool for many graph mining

Woojeong Jin
Seoul National University
E-mail: woojung211@snu.ac.kr

Jinhong Jung
Seoul National University
E-mail: jinhongjung@snu.ac.kr

U Kang
Seoul National University
E-mail: ukang@snu.ac.kr

Table 1: Comparison of our proposed SuRe and existing methods with respect to various aspects in ranking and link prediction tasks. SuRe outperforms all learning methods in terms of accuracy, speed, scalability, and memory usage.

Method	**Supervised?**	**Accuracy**	**Speed**	**Scalability**	**Memory Usage**
QUINT [14]	Yes	High	Low	Low	High
SRW [5]	Yes	High	Low	Low	Low
RWR [7]	No	Low	High	High	Low
SuRe	**Yes**	**Higher**	**High**	**High**	**Low**

applications [1, 3, 2, 12, 5]. Among various relevance measures, Random Walk with Restart (RWR) [7] provides useful node-to-node relevance scores by considering global network structure [8] and intricate edge relationships [25]. RWR has been successfully exploited in a wide range of data mining applications such as ranking [27, 23, 10], link prediction [1, 3, 2, 14, 5], community detection [4, 29], anomaly detection [24], etc.

However, RWR has two challenges for providing more effective relevance scores. First, RWR assumes a fixed restart probability on all nodes, i.e., a random surfer jumps back to the query node with the same probability regardless of where the surfer is located. This assumption prevents the surfer from considering the query node's preferences for other nodes, thereby limiting the expressiveness of random walk for measuring good relevance scores. Second, RWR requires users to heuristically select the restart probability parameter without strong theoretical justification to choose the parameter.

In this paper, we propose a novel relevance measure Random Walk with Extended Restart (RWER), an extended version of RWR, which reflects a query node's preferences on relevance scores by allowing a distinct restart probability for each node. We also propose a supervised learning method SuRe (Supervised Restart for RWER) that automatically finds optimal restart probabilities in RWER from a given graph. Extensive experiments show that our method provides the best link prediction accuracy: e.g., SuRe boosts MAP (Mean Average Precision) by up to 14.7% on the best competitor as shown in Figure 1. Table 1 summarizes strength of SuRe compared to existing methods. Our main contributions are summarized as follows:

- **Model.** We propose Random Walk with Extended Restart (RWER), a new random walk model to improve the expressiveness of RWR. RWER allows each node to have a distinct restart probability so that the random surfer has a finer control on preferences for each node.
- **Learning.** We propose SuRe, an algorithm for learning the restart probabilities in RWER from data. SuRe automatically determines the best restart probabilities.
- **Experiment.** We empirically demonstrate that our proposed method improves accuracy in all dataset. Specifically, our proposed method improves MAP by up to 14.7% and Precision@20 by up to 10.1% on the best competitor.

The source code of our method and datasets used in this paper are available at `http://datalab.snu.ac.kr/sure`. The rest of this paper is organized as follows. Section 2 presents a preliminary on RWR and defines the problem. Our proposed

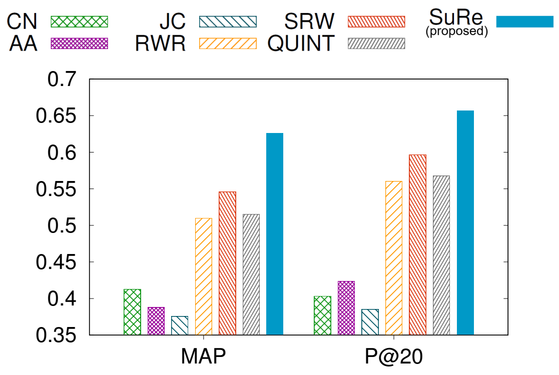


Fig. 1: Link prediction performance on the HepTh dataset. SuRe shows the highest accuracies: 14.7% higher MAP, and 10.1% higher Precision@20 compared to the best existing method.

methods are described in Section 3. After presenting experimental results in Section 4, we provide a review on related works in Section 5. Lastly, we conclude in Section 6.

2 Preliminaries

In this section, we describe the preliminaries on Random Walk with Restart. Then, we formally define the problem handled in this paper. We use A_{ij} or $A(i,j)$ to denote the entry at the intersection of the i-th row and j-th column of matrix $\mathbf{A}$, $\mathbf{A}(i,:)$ to denote the i-th row of $\mathbf{A}$, and $\mathbf{A}(:,j)$ to denote the j-th column of $\mathbf{A}$. The i-th element of the vector $\mathbf{x}$ is denoted by x_i.

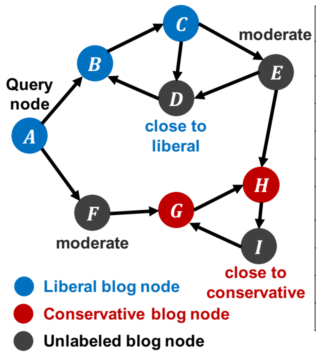
2.1 Random Walk with Restart.

Random walk with restart (also known as Personalized PageRank, PPR with a single seed node) [27] measures each node's proximity (relevance) w.r.t. a given query node s in a graph. RWR assumes a random surfer who starts at node s. The surfer moves to one of its neighboring nodes with probability $1-c$ or restarts at node s with probability c. When the surfer moves from u to one of its neighbors, each neighbor v is selected with a probability proportional to the weight in the edge (u,v). The relevance score between seed node s and node u is the stationary probability that the surfer is at node u. If the score is large, we consider that nodes s and u are highly related.

Limitations. RWR cannot consider a query node's preferences for estimating relevance scores between the query node and other nodes. For example, suppose we compute relevance scores from the query node A to other nodes in a political blog network in Figure 2 where blue colored nodes (A, B, and C) are liberal blogs, red colored ones (G and H) are conservative, black ones (D, E, F, and I) are not labeled, and an edge between nodes indicates a hyperlink between the corresponding blogs. Based on the topology of the graph, we consider that nodes E and F tend to be moderate, node D is likely to be liberal, and node I is likely to be conservative. Since the query node A is a liberal blog, node A will prefer other liberal nodes to conservative nodes. However, a conservative node G is ranked higher than nodes related to liberal blogs such as nodes C and D in the ranking result of RWR as shown in the left table of Figure 2. The reason is that preferences

Table 2: Table of symbols.

Symbol	Definition
$\mathcal{G}$	input graph
n	number of nodes in $\mathcal{G}$
m	number of edges in $\mathcal{G}$
s	query node (= seed node)
c	restart probability
$\mathbf{c}$	$(n \times 1)$ restart probability vector
$\mathbf{r}$	$(n \times 1)$ relevance vector
$\mathbf{o}$	$(n \times 1)$ origin vector
$\mathbf{A}$	$(n \times n)$ adjacency matrix of $\mathcal{G}$
$\tilde{\mathbf{A}}$	$(n \times n)$ row-normalized matrix of $\mathcal{G}$
$\mathbf{A}(i,:)$	$(1 \times n)$ i-th row of a matrix $\mathbf{A}$
$\mathbf{A}(:,j)$	$(n \times 1)$ j-th column of a matrix $\mathbf{A}$
$\mathbf{J}^{ij}$	$(n \times n)$ single-entry matrix whose (i,j) entry is 1
$\mathbf{q}$	$(n \times 1)$ starting vector
P, N	set of positive and negative nodes
$\circ$	Hadamard product



RWR			RWER & SuRe (proposed)		
Rank	Node $c = 0.1$	Node $c = 0.5$	Rank	Node	c
1	G	A	1	A	0.12
2	H	B	2	B	0.06
3	I	F	3	C	0.09
4	A	G	4	D	0.19
5	B	C	5	F	0.48
6	C	H	6	E	0.21
7	D	D	7	G	0.94
8	E	I	8	H	0.83
9	F	E	9	I	0.85

▪ c is a restart probability

Fig. 2: Example of RWR and our proposed approaches RWER & SuRe on a political blog network. Blue colored nodes (A, B, and C) are liberal blogs, red colored ones (G and H) are conservative, and black colored ones (D, E, F, and I) are unlabeled blogs. RWR uses the fixed restart probability **0.1** or **0.5** while our proposed RWER uses distinct restart probabilities on nodes. Note that RWR shows different ranking results depending on the restart probability as shown in the left table. The ranking result by our proposed RWER and SuRe is more desirable for the query node A (liberal) than those of RWR because many liberal blog nodes are ranked high in the ranking result as shown in the right table.

are not considered in RWR, and the random surfer jumps back to the query node A with a fixed restart probability c wherever the surfer is. On the other hand, RWER reflects the query node's preferences on relevance scores by allowing a distinct restart probability for each node as shown in the right table of Figure 2.

Another practical problem is that it is non-trivial to set an appropriate value of the restart probability c for different applications since we need to manually choose c so that the restart probability provides optimal relevance scores for each application. RWR scores are highly affected by the restart probability; the ranking results of each restart probability ($c = 0.1$ and $c = 0.5$) are quite different as seen in the left table of Figure 2. In contrast, our learning method SuRe automatically determines the optimal restart probabilities for all nodes based on the query node's preferences as well as relationships between nodes. The detailed descriptions of our proposed approaches RWER and SuRe are presented in Section 3.

2.2 Problem Definition.

We are given a graph $\mathcal{G}$ with n nodes and m edges, a query node s, and side information from the query node. The side information contains a set of *positive* nodes $P = \{x_1, ..., x_k\}$ that s prefers, and a set of *negative* nodes $N = \{y_1, ..., y_l\}$ that s dislikes. Our task is to learn restart probabilities for all nodes such that relevance scores of the positive nodes are greater than those of the negative ones.

3 Proposed Method

In this section, we first describe Random Walk with Extended Restart (RWER), our proposed model for extended restart probabilities. We then propose SuRe, an efficient algorithm for learning the restart probabilities.

3.1 Overview of Random Walk with Extended Restart.

RWER is a novel relevance model reflecting a query node's preferences on relevance scores. The main idea of RWER is that we introduce a restart probability *vector* each of whose entry corresponds to a restart probability at a node, so that the restart probabilities are related to the preferences for the nodes.

In RWER, a restart probability of each node is interpreted as the degree of boredom of a node w.r.t. the query node. That is, if the restart probability on a node is large, then the surfer runs away from the current node to the query node (i.e., the surfer becomes bored at the node). On the other hand, if the restart probability of the node is small, then surfer desires to move around the node's neighbors (i.e., the surfer has an interest in the node and its neighbors).

As depicted in Figure 2, each node has its own restart probability in our model RWER. The restart probabilities are determined by our supervised learning method SuRe (Section 3.5) from the query (liberal) node A, the positive (liberal) nodes B and C, and the negative (conservative) nodes G and H. Note that a ranking list where many liberal nodes are ranked high is desirable for the query node A since A is liberal. As shown in Figure 2, using distinct restart probabilities for each node by RWER provides more satisfactory rankings for the query node than using a single restart probability for all nodes by RWR. The restart probabilities of liberal nodes are smaller than those of conservative nodes, which implies that the random surfer prefers searching around the liberal nodes such as B and C while the surfer is likely to run away from the conservative nodes such as G and H.

One might think that it is enough to simply assign small restart probabilities to positive nodes and large restart probabilities to negative nodes for a desirable ranking. However, the restart probabilities should be determined also for unlabeled nodes, and the probabilities should reflect intricate relationships between nodes as well as the query node's preferences. For example, the restart probability of node F in Figure 2 is relatively moderate because node F is located between a liberal node A and a conservative node G. Also, the restart probability of node D is small since node D is close to other liberal nodes B and C. Similarly, the restart probability of node I is large since node I is closely related to other conservative nodes G and H.

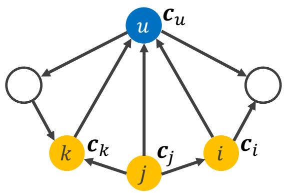


Fig. 3: Example of a network. Each node has its own restart probability.

3.2 Formulation of Random Walk with Extended Restart.

We formulate RWER in this section. We first explain the formulation using the example shown in Figure 3, and present general equations. In the example, the surfer goes to one of its neighbors or jumps back to the query node. To obtain the RWER probability r_u at time $t+1$, we should take into account the scores of the three nodes which are i, j and k at time t. Suppose the surfer is at the node i at time t. The surfer can go to an out-neighbor through one of the two outgoing edges with probability $1-c_i$. Note that every node has a distinct restart probability and node i has a restart probability c_i in this case. Without the restart action, $r_u^{(t+1)}$ in Figure 3 is defined as follows:

$$r_u^{(t+1)} \leftarrow (1-c_i)\frac{r_i^{(t)}}{2} + (1-c_j)\frac{r_j^{(t)}}{3} + (1-c_k)r_k^{(t)}$$

Also, the surfer on any node v jumps back to the query node with probability c_v. The above equation is rewritten as follows considering the restart action of the random surfer: -3mm

$$\begin{aligned} r_u^{(t+1)} \leftarrow & (1-c_i)\frac{r_i^{(t)}}{2} + (1-c_j)\frac{r_j^{(t)}}{3} + (1-c_k)r_k^{(t)} \\ & + \left(c_1 r_1^{(t)} + \cdots + c_v r_v^{(t)} + \cdots + c_n r_n^{(t)}\right) 1(u=s) \end{aligned}$$

where $1(u=s)$ is 1 if u is the query node s; otherwise, it is 0. Note that the restart term is different from that of the traditional random walk with restart.

Based on the aforementioned example, the recursive equation of our model is defined as follows:

$$r_u = \left(\sum_{v \in \mathbf{IN}_u} (1-c_v)\frac{r_v}{|\mathbf{OUT}_v|}\right) + \left(\sum_v c_v r_v\right) 1(u=s) - 2mm \qquad (1)$$

where $\mathbf{IN}_i$ is the set of in-neighbors of node i, and $\mathbf{OUT}_i$ is the set of out-neighbors of node i.

Equation (1) is expressed in the form of a matrix equation as follows:

$$\mathbf{r} = \tilde{\mathbf{A}}^\top (\mathbf{I} - diag(\mathbf{c}))\mathbf{r} + \left(\mathbf{c}^\top \mathbf{r}\right) \mathbf{q} \qquad (2)$$

where $\tilde{\mathbf{A}}$ is a row-normalized matrix of the adjacency matrix $\mathbf{A}$, $\mathbf{c}$ is a restart vector whose i-th entry is c_i, $diag(\mathbf{c})$ is a matrix whose $diag(\mathbf{c})_{ii} = c_i$ and other entries are 0, and $\mathbf{q}$ is a vector whose s-th element is 1 and all other elements are

0. Notice that if $\mathbf{c}$ is a vector all of whose elements are the same, then the RWER is equal to RWR (or PPR).

The following lemma shows that equation (2) can be represented as a closed form equation.

Lemma 1 *The closed form w.r.t.* $\mathbf{r}$ *in equation* (2) *is represented as follows:*

$$\mathbf{r} = (\mathbf{I} - \mathbf{B})^{-1}\mathbf{q} \tag{3}$$

where $\mathbf{B}$ *is* $\tilde{\mathbf{A}}^\top(\mathbf{I} - \mathit{diag}(\mathbf{c})) + \mathbf{q}(\mathbf{c} - \mathbf{1})^\top$, $\tilde{\mathbf{A}}$ *is a row-normalized matrix, and* $\mathbf{1}$ *is an all-ones vector.*

Proof *Note that* $\mathbf{c}^\top\mathbf{r}$ *is a scalar; thus,* $(\mathbf{c}^\top\mathbf{r})\mathbf{q} = \mathbf{q}(\mathbf{c}^\top\mathbf{r}) = (\mathbf{q}\mathbf{c}^\top)\mathbf{r}$. *Hence, equation* (2) *is represented as follows:*

$$\begin{aligned}\mathbf{r} &= \tilde{\mathbf{A}}^\top(\mathbf{I} - \mathit{diag}(\mathbf{c}))\mathbf{r} + \left(\mathbf{q}\mathbf{c}^\top\right)\mathbf{r} \\ &= \tilde{\mathbf{A}}^\top(\mathbf{I} - \mathit{diag}(\mathbf{c}))\mathbf{r} + \left(\mathbf{q}\mathbf{c}^\top\right)\mathbf{r} - \mathbf{q} + \mathbf{q}\end{aligned}$$

Since $\mathbf{r}$ *is a probability vector and* $\mathbf{1}^\top\mathbf{r} = 1$, *the above equation is written in the following equation:*

$$\begin{aligned}\mathbf{r} &= \tilde{\mathbf{A}}^\top(\mathbf{I} - \mathit{diag}(\mathbf{c}))\mathbf{r} + \left(\mathbf{q}\mathbf{c}^\top\right)\mathbf{r} - \mathbf{q}\left(\mathbf{1}^\top\mathbf{r}\right) + \mathbf{q} \\ &= \tilde{\mathbf{A}}^\top(\mathbf{I} - \mathit{diag}(\mathbf{c}))\mathbf{r} + \left(\mathbf{q}(\mathbf{c} - \mathbf{1})^\top\right)\mathbf{r} + \mathbf{q} \\ &= \left(\tilde{\mathbf{A}}^\top(\mathbf{I} - \mathit{diag}(\mathbf{c})) + \mathbf{q}(\mathbf{c} - \mathbf{1})^\top\right)\mathbf{r} + \mathbf{q} \\ &= \mathbf{B}\mathbf{r} + \mathbf{q}\end{aligned}$$

where $\mathbf{B}$ *is* $\tilde{\mathbf{A}}^\top(\mathbf{I} - \mathit{diag}(\mathbf{c})) + \mathbf{q}(\mathbf{c} - \mathbf{1})^\top$. *Finally,* $\mathbf{r}$ *is represented in the following closed form:*

$$\mathbf{r} = (\mathbf{I} - \mathbf{B})^{-1}\mathbf{q}$$

Note that $\mathbf{I} - \mathbf{B}$ *is invertible (see the following Lemma 2).* ■

Lemma 2 *Suppose* $\mathbf{B}$ *is* $\tilde{\mathbf{A}}^\top(\mathbf{I} - \mathit{diag}(\mathbf{c})) + \mathbf{q}(\mathbf{c} - \mathbf{1})^\top$. *Then,* $\mathbf{M} = \mathbf{I} - \mathbf{B}$ *is invertible.*

Proof *Let* $\mathbf{C}$ *be* $\mathbf{I} - \mathit{diag}(\mathbf{c})$, *and* $\mathbf{X}$ *be* $\mathbf{I} - \tilde{\mathbf{A}}^\top\mathbf{C}$. *Then,* $\mathbf{M} = \mathbf{I} - \mathbf{B}$ *is represented as follows:*

$$\begin{aligned}\mathbf{M} &= \mathbf{I} - \tilde{\mathbf{A}}^\top\mathbf{C} - \mathbf{q}(\mathbf{c} - \mathbf{1})^\top \\ &= \mathbf{X} + \mathbf{q}(\mathbf{1} - \mathbf{c})^\top\end{aligned}$$

Note that $(\tilde{\mathbf{A}}^\top\mathbf{C})_{ij} \geq 0$, $1 \leq i, j \leq n$ *and the largest eigenvalue* $\lambda_{max}(\tilde{\mathbf{A}}^\top\mathbf{C}) \leq 1$ *since* $\tilde{\mathbf{A}}$ *is stochastic and* $\mathbf{C}$ *is sub-stochastic. In other words,* $\mathbf{X} = \mathbf{I} - \tilde{\mathbf{A}}^\top\mathbf{C}$ *is M-matrix [6]; thus,* $\mathbf{X}^{-1}$ *exists and all entries of* $\mathbf{X}^{-1}$ *are nonnegative. By Sherman-Morrison lemma [22],* $(\mathbf{X} + \mathbf{q}(\mathbf{1} - \mathbf{c})^\top)^{-1}$ *is represented as follows:*

$$\left(\mathbf{X} + \mathbf{q}(\mathbf{1} - \mathbf{c})^\top\right)^{-1} = \mathbf{X}^{-1} - \frac{\mathbf{X}^{-1}\mathbf{q}(\mathbf{1} - \mathbf{c})^\top\mathbf{X}^{-1}}{1 + (\mathbf{1} - \mathbf{c})^\top\mathbf{X}^{-1}\mathbf{q}}$$

Algorithm 1 Normalization phase of RWER

```
Input: adjacency matrix $\mathbf{A}$
Output: row-normalized matrix $\tilde{\mathbf{A}}$
1: compute a degree diagonal matrix $\mathbf{D}$ of $\mathbf{A}$ (i.e., $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$)
2: compute a normalized matrix, $\tilde{\mathbf{A}} = \mathbf{D}^{-1}\mathbf{A}$.
3: return $\tilde{\mathbf{A}}$
```

Algorithm 2 Iteration phase of RWER

```
Input: row-normalized matrix $\tilde{\mathbf{A}}$, query node $s$, restart probability vector $\mathbf{c}$, and error tolerance $\epsilon$
Output: RWER score vector $\mathbf{r}$
1: set the starting vector $\mathbf{q}$ from the seed node $s$
2: repeat
3:     $\mathbf{r}' \leftarrow \tilde{\mathbf{A}}^\top (\mathbf{I} - diag(\mathbf{c}))\, \mathbf{r} + (\mathbf{c}^\top \mathbf{r})\, \mathbf{q}$
4:     compute error, $\delta = \|\mathbf{r}' - \mathbf{r}\|$
5:     update $\mathbf{r} \leftarrow \mathbf{r}'$
6: until $\delta < \epsilon$
7: return $\mathbf{r}$
```

Since all entries of $\mathbf{X}^{-1}$, $\mathbf{1} - \mathbf{c}$ *and* $\mathbf{q}$ *are nonnegative,* $(\mathbf{1} - \mathbf{c})^\top \mathbf{X}^{-1}\mathbf{q} \geq 0$*; thus,* $1 + (\mathbf{1} - \mathbf{c})^\top \mathbf{X}^{-1}\mathbf{q} \geq 1$. *Hence, the right side of the above equation exists, i.e.,* $\left(\mathbf{X} + \mathbf{q}(\mathbf{1} - \mathbf{c})^\top\right)^{-1} = \mathbf{M}^{-1}$ *exists.* $\mathbf{I} - \mathbf{B}^\top$ *is also invertible, which is proved similarly to the proof of* $\mathbf{I} - \mathbf{B}$. ■

Note that if $\mathbf{c}$ is given, the RWER vector $\mathbf{r}$ can be calculated using the closed form in Lemma 1. However, the computation using the closed form requires $O(n^3)$ time and $O(n^2)$ memory space due to the matrix inversion where n is the number of nodes; thus, this approach is impractical when we need to compute RWER scores in large-scale graphs. In order to avoid the heavy computational cost, we exploit an efficient iterative algorithm described in Section 3.3.

3.3 Algorithm for Random Walk with Extended Restart.

We present an iterative algorithm for computing RWER scores efficiently. Our algorithm is based on power iteration and comprises two phases: a normalization phase (Algorithm 1) and an iteration phase (Algorithm 2).

Normalization phase (Algorithm 1). Our proposed algorithm first computes the out-degree diagonal matrix $\mathbf{D}$ of $\mathbf{A}$ (line 1). Then, the algorithm computes the row normalized matrix $\tilde{\mathbf{A}}$ using $\mathbf{D}$ (line 2).

Iteration phase (Algorithm 2). Our algorithm computes the RWER score vector $\mathbf{r}$ for the seed node s in the iteration phase. As described in Section 3.2, the vector $\mathbf{q}$ denotes a length-n starting vector whose entry at the index of the seed node is 1 and otherwise 0 (line 1). Our algorithm iteratively computes equation (2) (line 3). We then compute the error δ between $\mathbf{r}'$, the result in the previous iteration, and $\mathbf{r}$ (line 4). Next, we update $\mathbf{r}'$ into $\mathbf{r}$ for the next iteration (line 5). The iteration stops when the error δ is smaller than a threshold ϵ (line 6).

Theoretical analysis. We analyze the convergence of the iterative algorithm in Theorem 1 and the time complexity in Theorem 2. We assume that all the matrices considered are saved in a sparse format, such as the compressed column storage [19], which stores only non-zero entries, and that all the matrix operations exploit such sparsity by only considering non-zero entries.

Theorem 1 (Convergence) *Suppose the graph represented by* $\tilde{\mathbf{A}}$ *is irreducible and aperiodic. Then, the power iteration algorithm (Algorithm 2) for* RWER *converges.*

Proof Equation (2) *is represented as follows:*

$$\begin{aligned}\mathbf{r} &= \tilde{\mathbf{A}}^{\top}(\mathbf{I} - diag(\mathbf{c}))\mathbf{r} + \left(\mathbf{q}\mathbf{c}^{\top}\right)\mathbf{r} \\ &= \left(\tilde{\mathbf{A}}^{\top}(\mathbf{I} - diag(\mathbf{c})) + \left(\mathbf{q}\mathbf{c}^{\top}\right)\right)\mathbf{r} = \mathbf{G}\mathbf{r}\end{aligned}$$

Note that $\mathbf{G}$ *is a column stochastic matrix. Moreover,* $\mathbf{G}$ *is irreducible since* $\tilde{\mathbf{A}}$ *is irreducible, and aperiodic due to the self-loop at node* s *by restart. Hence,* $\mathbf{r}$ *is the eigenvector corresponding to the principal eigenvalue of* $\mathbf{G}$*, and the power iteration for* $\mathbf{r}$ *converges [13].*

Theorem 2 (Time Complexity) *The time complexity of Algorithm 2 is* $O(Tm)$ *where* T *is the number of iterations, and* m *is the number of edges.*

Proof We assume that the number of edges is greater than that of nodes for simplicity. The iterative algorithm for computing RWER *scores takes* $O(Tm)$ *time since each iteration requires the sparse matrix-vector multiplication which takes* $O(m)$ *time.*

Theorem 2 indicates that our method in Algorithm 2 presents the linear scalability w.r.t the number of edges.

3.4 Cost Function.

Although our relevance measure RWER improves the expressiveness of RWR by introducing a distinct restart probability for each node, it is difficult to manually investigate the optimal restart probabilities for all nodes in large graphs. In this section, we define the cost function for finding the optimal restart probabilities.

As mentioned in Section 2.2, our goal is to set the optimal restart probabilities so that the relevance scores of positive nodes outweigh those of negative nodes. We define the following cost function:

$$\arg\min_{\mathbf{c}} F(\mathbf{c}) = \lambda \|\mathbf{c} - \mathbf{o}\|^2 + \sum_{x \in P, y \in N} h(r_y - r_x) \tag{4}$$

where λ is a regularization parameter that controls the importance of the regularization term, $\mathbf{o}$ is a given origin vector, h is a loss function, and r_x and r_y are RWER scores of nodes x and y, respectively. The cost function is obtained from the pairwise differences between the RWER scores of positive and negative nodes. Given an increasing loss function h, $F(\mathbf{c})$ is minimized as the scores of positive nodes are maximized and those of negative nodes are minimized. The origin vector $\mathbf{o}$ prevents the $\mathbf{c}$ vector becoming too small, and serves as a model regularizer which helps avoid overfitting and thus improves accuracy, as we will see in Section 4.4. We set $\mathbf{o}$ to a constant vector all of whose elements are set to a constant. We use the loss function $h(x) = (1 + \exp(-x/b))^{-1}$ since the loss function maximizes AUC [16, 5].

3.5 SuRe - Optimizing the Cost Function.

Our goal is to minimize equation (4) with respect to $\mathbf{c}$. Note that the objective function $F(\mathbf{c})$ is not convex. Thus, we exploit the gradient descent method to find the local minimum of function $F(\mathbf{c})$. For the purpose, we first need to obtain the derivative of $F(\mathbf{c})$ w.r.t. $\mathbf{c}$:

$$\begin{aligned}\frac{\partial F(\mathbf{c})}{\partial \mathbf{c}} &= 2\lambda(\mathbf{c}-\mathbf{o}) + \sum_{x\in P, y\in N} \frac{\partial h(r_y - r_x)}{\partial \mathbf{c}} \\ &= 2\lambda(\mathbf{c}-\mathbf{o}) + \sum_{x\in P, y\in N} \frac{\partial h(\delta_{yx})}{\partial \delta_{yx}}\left(\frac{\partial r_y}{\partial \mathbf{c}} - \frac{\partial r_x}{\partial \mathbf{c}}\right)\end{aligned} \tag{5}$$

where δ_{yx} is $r_y - r_x$. The derivative $\frac{\partial h(\delta_{yx})}{\partial \delta_{yx}}$ of the loss function is $\frac{1}{b}h(\delta_{yx})(1 - h(\delta_{yx}))$.

In order to obtain the derivative $\frac{\partial r_x}{\partial \mathbf{c}}$, we have to calculate the derivative of the relevance score r_x w.r.t. c_i which is the i-th element of $\mathbf{c}$. Let $\mathbf{M}$ be $(\mathbf{I}-\mathbf{B})^{-1}$; then, $\mathbf{r} = (\mathbf{I}-\mathbf{B})^{-1}\mathbf{q} = \mathbf{M}\mathbf{q}$, $\mathbf{M}(:,s) = \mathbf{r}$, and $M(x,s) = r_x$, from Theorem 1.

Since $\mathbf{M}$ is the inverse of $\mathbf{I}-\mathbf{B}$, according to [18], $\frac{\partial \mathbf{M}}{\partial c_i}$ becomes:

$$\frac{\partial \mathbf{M}}{\partial c_i} = -\mathbf{M}\frac{\partial(\mathbf{I}-\mathbf{B})}{\partial c_i}\mathbf{M} = \mathbf{M}(-\tilde{\mathbf{A}}^\top \mathbf{J}^{ii} + \mathbf{J}^{si})\mathbf{M}$$

where $\mathbf{J}^{ij}$ is a single-entry matrix whose (i,j)th entry is 1 and all other elements are 0. Based on the above equation, $\frac{\partial M(x,s)}{\partial c_i}$ is represented as follows:

$$\frac{\partial M(x,s)}{\partial c_i} = \mathbf{M}(x,:)(-\tilde{\mathbf{A}}^\top(:,i) + \mathbf{e}_s)M(i,s)$$

where $\mathbf{e}_s$ is a length n unit vector whose s-th entry is 1. Note that $\frac{\partial M(x,s)}{\partial c_i}$ is calculated for $1 \le i \le n$; then, $\frac{\partial r_x}{\partial \mathbf{c}}$ is written in the following equation:

$$\frac{\partial r_x}{\partial \mathbf{c}} = \frac{\partial M(x,s)}{\partial \mathbf{c}} = \left((-\tilde{\mathbf{A}} + \mathbf{1}\mathbf{e}_s^\top)\mathbf{M}(x,:)^\top\right) \circ \mathbf{M}(:,s) \tag{6}$$

where $\circ$ denotes Hadamard product, and $\mathbf{1}$ is an all-ones vector of length n. Similarly, $\frac{\partial r_y}{\partial \mathbf{c}}$ is calculated by switching x to y. Using the equation (6), $\sum_{x\in P, y\in N} \frac{\partial h(\delta_{yx})}{\partial \delta_{yx}}(\frac{\partial r_y}{\partial \mathbf{c}} - \frac{\partial r_x}{\partial \mathbf{c}})$ in equation (5) is represented as follows:

$$\begin{aligned}&\left((-\tilde{\mathbf{A}} + \mathbf{1}\mathbf{e}_s^\top)\sum_{x\in P, y\in N}\frac{\partial h(\delta_{yx})}{\partial \delta_{yx}}(\mathbf{M}(y,:) - \mathbf{M}(x,:))^\top\right) \circ \mathbf{M}(:,s) \\ &= \left((-\tilde{\mathbf{A}} + \mathbf{1}\mathbf{e}_s^\top)\tilde{\mathbf{r}}\right) \circ \mathbf{r}\end{aligned} \tag{7}$$

where $\mathbf{r} = \mathbf{M}(:,s)$ is an RWER score vector, and $\tilde{\mathbf{r}} = \sum_{x\in P, y\in N}\frac{\partial h(\delta_{yx})}{\partial \delta_{yx}}(\mathbf{M}(y,:) - \mathbf{M}(x,:))^\top$. Then, $\frac{\partial F(\mathbf{c})}{\partial \mathbf{c}}$ in equation (5) is represented as follows:

$$\frac{\partial F(\mathbf{c})}{\partial \mathbf{c}} = 2\lambda(\mathbf{c}-\mathbf{o}) + \left((-\tilde{\mathbf{A}} + \mathbf{1}\mathbf{e}_s^\top)\tilde{\mathbf{r}}\right) \circ \mathbf{r} \tag{8}$$

Algorithm 3 SuRe - Learning a restart vector $\mathbf{c}$

```
Input: adjacency matrix $\mathbf{A}$, query node $s$, positive set $P$, negative set $N$, origin vector $\mathbf{o}$,
    parameter $b$ of loss function $h$, and the learning rate $\eta$
Output: the learned restart vector $\mathbf{c}$
1: randomly initialize $\mathbf{c}$
2: while $\mathbf{c}$ does not converge do
3:     compute $\mathbf{r} = \mathbf{M}(:, s)$ based on equation (2) (Algorithm 2)
4:     compute $\tilde{\mathbf{r}} = \sum_{x\in P, y\in N} \frac{\partial h(\delta_{yx})}{\partial \delta_{yx}} (\mathbf{M}(y,:) - \mathbf{M}(x,:))^\top$ by a linear system solver
       (Lemma 3)
5:     compute $\frac{\partial F(\mathbf{c})}{\partial \mathbf{c}}$ by equation (8)
6:     update $\mathbf{c} \leftarrow \mathbf{c} - \eta \frac{\partial F(\mathbf{c})}{\partial \mathbf{c}}$
7: end while
8: return the learned restart vector $\mathbf{c}$
```

Notice that we do not obtain $\mathbf{M}$ explicitly to compute $\mathbf{M}(:, s)$ in equations (8) since $\mathbf{M}$ is the inverse of $\mathbf{I} - \mathbf{B}$ and inverting a large matrix is infeasible as mentioned in Section 3.2. Instead, we use the iterative method described in Algorithm 2 to get $\mathbf{r} = \mathbf{M}(:, s)$. However, the problem is that we also require rows of $\mathbf{M}$ (i.e., $\mathbf{M}(x, :)$ in $\tilde{\mathbf{r}}$), and Algorithm 2 only computes a column of $\mathbf{M}$ for a given seed node. How can we calculate $\tilde{\mathbf{r}}$ without inverting $\mathbf{M}$? $\tilde{\mathbf{r}}$ is computed iteratively by the following lemma:

Lemma 3 *From the result of equation* (7), $\tilde{\mathbf{r}} = \sum_{x\in P, y\in N} \frac{\partial h(\delta_{yx})}{\partial \delta_{yx}} (\mathbf{M}(y,:) - \mathbf{M}(x,:))^\top$ *which is represented as* $\tilde{\mathbf{r}} = \mathbf{M}^\top \tilde{\mathbf{p}} \Leftrightarrow (\mathbf{I} - \mathbf{B}^\top)\tilde{\mathbf{r}} = \tilde{\mathbf{p}}$ *where* $\tilde{\mathbf{p}} = \sum_{x\in P, y\in N} \frac{\partial h(\delta_{yx})}{\partial \delta_{yx}} (\mathbf{e}_y - \mathbf{e}_x)$*,* $\mathbf{e}_x$ *is an* $n \times 1$ *vector whose* x*-th element is* 1 *and the others are* 0*, and* $\delta_{yx} = r_y - r_x$*. Then,* $\tilde{\mathbf{r}}$ *is the solution of the linear system* $(\mathbf{I} - \mathbf{B}^\top)\tilde{\mathbf{r}} = \tilde{\mathbf{p}}$ *which is solved by an iterative method for linear systems.*

Proof $\mathbf{M}(x, :)^\top$ *is a column vector which is the transpose of the* x*-th row of the matrix* $\mathbf{M}$*. In other words,* $\mathbf{M}(x, :) = \mathbf{e}_x^\top \mathbf{M} \Leftrightarrow \mathbf{M}(x, :)^\top = \mathbf{M}^\top \mathbf{e}_x$ *where* $\mathbf{e}_x$ *is an* $n \times 1$ *vector whose* x*-th element is* 1 *and the others are* 0*. Then,* $\tilde{\mathbf{r}}$ *is represented as follows:*

$$\begin{aligned}
\tilde{\mathbf{r}} &= \sum_{x\in P, y\in N} \frac{\partial h(\delta_{yx})}{\partial \delta_{yx}} (\mathbf{M}(y,:) - \mathbf{M}(x,:))^\top \\
&= \sum_{x\in P, y\in N} \frac{\partial h(\delta_{yx})}{\partial \delta_{yx}} \left(\mathbf{M}^\top \mathbf{e}_y - \mathbf{M}^\top \mathbf{e}_x\right) \\
&= \mathbf{M}^\top \sum_{x\in P, y\in N} \frac{\partial h(\delta_{yx})}{\partial \delta_{yx}} (\mathbf{e}_y - \mathbf{e}_x) = \mathbf{M}^\top \tilde{\mathbf{p}}
\end{aligned}$$

where $\tilde{\mathbf{p}} = \sum_{x\in P, y\in N} \frac{\partial h(\delta_{yx})}{\partial \delta_{yx}} (\mathbf{e}_y - \mathbf{e}_x)$*. Also,* $\tilde{\mathbf{r}}$ *is represented as the following linear system:*

$$\tilde{\mathbf{r}} = \mathbf{M}^\top \tilde{\mathbf{p}} = (\mathbf{I} - \mathbf{B})^{-\top} \tilde{\mathbf{p}} \Leftrightarrow (\mathbf{I} - \mathbf{B}^\top)\tilde{\mathbf{r}} = \tilde{\mathbf{p}}$$

where $\mathbf{B} = \tilde{\mathbf{A}}^\top (\mathbf{I} - \mathit{diag}(\mathbf{c})) + \mathbf{q}(\mathbf{c} - \mathbf{1})^\top$*.*

∎

Note that $\mathbf{M}^{-\top} = \mathbf{I} - \mathbf{B}^\top$ is non-symmetric and invertible (Lemma 2); thus, any iterative method for a non-symmetric matrix can be used to solve for $\tilde{\mathbf{r}}$. We

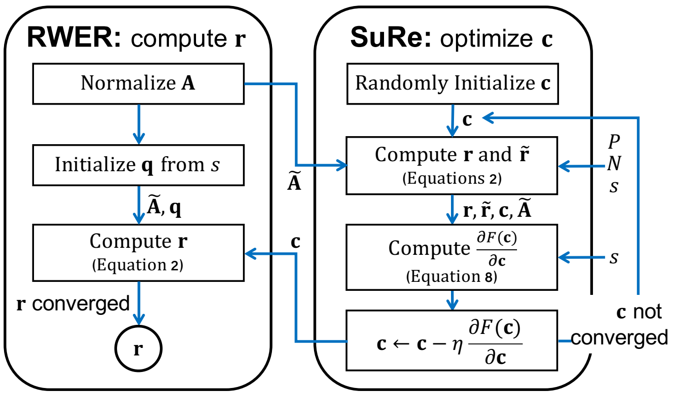


Fig. 4: Flowchart of RWER (Algorithms 1 and 2) and SuRe (Algorithm 3). SuRe learns restart probability vector **c**, and RWER computes our node relevance score vector **r** for a given seed node s.

use GMRES [20], an iterative method for solving linear systems since it is the state-of-the-art method in terms of efficiency and accuracy.

Optimization phase (Algorithm 3). SuRe algorithm for solving the optimization problem is summarized in Algorithm 3 and Figure 4. In the algorithm, we use a gradient-based method to update the restart probability **c** based on equation (8).

3.6 Theoretical analysis.

We analyze the time complexity of SuRe (Algorithm 3).

Lemma 4 *Let $|P|$ and $|N|$ denote the number of positive and negative nodes, respectively. The computation of $\tilde{\mathbf{r}} = \sum_{x,y} \frac{\partial h(\delta_{yx})}{\partial \delta_{yx}} (\mathbf{M}(y,:) - \mathbf{M}(x,:))^\top$ takes $O(Tm + |P||N|)$ time where T is the number of iterations until convergence, and m is the number of edges.*

Proof *Note that solving a sparse linear system $\mathbf{Ax} = \mathbf{b}$ with an iterative method such as GMRES [20] requires $O(T|\mathbf{A}|)$ time where T is the number of iterations, and $|\mathbf{A}|$ is the number of non-zeros of $\mathbf{A}$. Hence, it takes $O(T|\mathbf{I} - \mathbf{B}^\top|) = O(Tm)$ time to solve the linear system $(\mathbf{I} - \mathbf{B}^\top)\tilde{\mathbf{r}} = \tilde{\mathbf{p}}$ by the iterative method where the number of non-zeros of $\mathbf{I} - \mathbf{B}^\top$ is bounded by $O(m)$. In addition, setting $\tilde{\mathbf{p}}$ takes $O(|P||N|)$ time. Thus, the overall time complexity for computing $\sum_{x,y} \frac{\partial h(\delta_{yx})}{\partial \delta_{yx}} (\mathbf{M}(y,:) - \mathbf{M}(x,:))^\top$ is $O(Tm + |P||N|)$.* ■

Based on Lemma 4, the time complexity of Algorithm 3 is presented in Theorem 3.

Theorem 3 (Time complexity of Algorithm 3) *For a given graph with m non-zero elements, the learning algorithm* SuRe *takes $O(T_1(T_2 m + |P||N|))$ time where T_1 is the number of times **c** is updated with the gradient, and T_2 is the number of inner iterations for computing **r** and $\tilde{\mathbf{r}}$.*

Proof *The computation of **r** takes $O(T'm)$ time according to Theorem 3.2. The computation of $\tilde{\mathbf{r}} = \sum_{x,y} \frac{\partial h(\delta_{yx})}{\partial \delta_{yx}} (\mathbf{M}(y,:) - \mathbf{M}(x,:))^\top$ takes $O(T''m + |P||N|)$ time according to Lemma 4. Besides, the computation of $\frac{\partial F(\mathbf{c})}{\partial \mathbf{c}}$ takes additional*

Table 3: Dataset statistics. The query nodes are used for the ranking and the link prediction tasks.

Dataset	**#Nodes**	**#Edges**	**#Queries**
Wikipedia[1]	3,023,165	102,382,410	-
HepPh[1]	34,546	421,534	135
HepTh[1]	27,770	352,768	121
Polblogs[2]	1,490	19,025	115

[1] `http://konect.uni-koblenz.de`
[2] `http://www-personal.umich.edu/~mejn/netdata`

$O(m)$ time due to the sparse matrix-vector multiplication. Hence, SuRe *takes $O(T_1(T_2 m + |P||N|))$ time where T_1 is the number of iterations of the gradient descent procedure in order that the restart vector converges and $T_2 = T' + T''$.* ■

Theorem 3 implies that our learning method SuRe in Algorithm 3 provides linear time and space scalability w.r.t. the number m of edges. Notice that $|P|$ and $|N|$ are constants much smaller than m.

4 Experiment

We evaluate our proposed method with various baseline approaches. Since there is no ground-truth of node-to-node relevance scores in real-world graphs, we instead evaluate the performance of two representative applications based on relevance scores: ranking and link prediction. Based on these settings, we aim to answer the following questions:

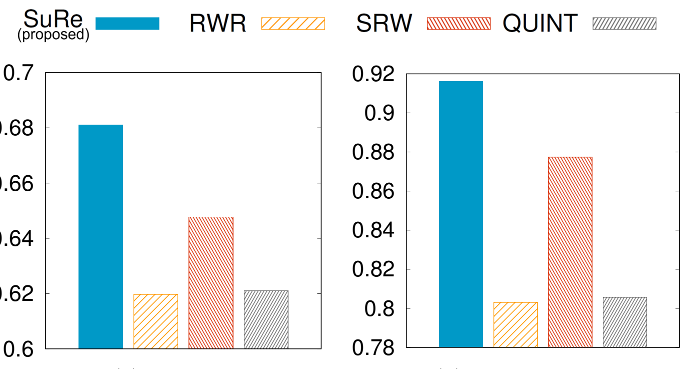
– **Q1. Ranking performance (Section 4.2).** Does our proposed method SuRe provide the best relevances scores for ranking compared to other methods?
– **Q2. Link prediction performance (Section 4.3).** How effective is SuRe for link prediction tasks?
– **Q3. Parameter sensitivity (Section 4.4).** How do parameters used in SuRe affect the accuracy of link prediction?
– **Q4. Effects of number of labeled nodes. (Section 4.5).** How does the number of labeled nodes affect the performance of link prediction?
– **Q5. Scalability (Section 4.6).** How well does SuRe scale up with the number of edges?

4.1 Experimental Settings.

Datasets. We experiment on various real-world network datasets. Datasets used in our experiments are summarized in Table 3. We use Polblogs for the ranking task (Section 4.2), HepPh and HepTh for the link prediction task (Section 4.3), and Wikipedia for the scalability experiment (Section 4.6). The Wikipedia dataset is a hyperlink network. The HepPh and HepTh datasets are collaboration networks where nodes are authors, and edges are collaboration relationships time-stamped from May 15, 1992 to August 14, 1996 and from October 1, 1993 to December 10, 1999, respectively. The Polblogs dataset is a political network made up of liberal and conservative blogs. Since only HepPh and HepTh have time information, we use them in the link prediction task. All experiments are performed on a Linux machine with Intel(R) Xeon E5-2630 v4 CPU @ 2.2GHz and 256GB memory.

Table 4: Ranking results of our proposed method SuRe and other methods w.r.t. a query node *obsidianwings*, a liberal blog. Red colored nodes are conservative blogs, and the black colored ones are liberal blogs. Our ranking result from SuRe contains only liberal nodes, indicating the best result, while other ranking results wrongly contain conservative nodes.

Rank	SuRe	RWR	SRW	QUINT
1	digbysb	**freerep**	digbysb	**freerep**
2	billmon	**michell**	tbogg	**michell**
3	gadflye	**prolife**	liberal	**prolife**
4	reachm	**rightwi**	billmon	**rightwi**
5	jameswo	digbysb	xnerg	digbysb
6	angrybe	**littleg**	corrent	**littleg**
7	marksch	billmon	**hughhew**	billmon
8	tbogg	jameswo	busybus	jameswo
9	wampum	reachm	pacific	reachm
10	stevegi	politic	nielsen	**hughhew**



(a) MAP (b) Precision@20

Fig. 5: Ranking performance on Polblogs. Our method SuRe provides the best ranking performance compared to existing methods in terms of MAP and Precision@20.

Methods. We compare our proposed method SuRe with Common Neighbor (CN) [15], Adamic-Adar (AA) [1], Jaccard's Coefficient (JC) [21], Random Walk with Restart (RWR) [7], Supervised Random Walks (SRW) [5], and QUINT [14].

Parameters. We set the origin vector **o** in SuRe and restart probability c in RWR, SRW, and QUINT to the ones that give the best performance. Also, in SuRe, SRW, and QUINT, we set $\lambda = 1$ among $\{0.01, 0.1, 1, 10\}$, and $b = 10^{-2}$ among $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ where the selected parameters give the best performance. For SRW, we use each node's degree and the number of common neighbors as features.

Evaluation Metrics. To compare the methods, we use Mean Average Precision (MAP), Area under the ROC curve (AUC), and Precision@20. MAP is the mean of average precisions for multiple queries. AUC is the expectation that a uniformly drawn random positive is ranked higher than a uniformly drawn random negative. Precision@20 is the precision at the top-20 position in a ranking result. The higher the values of the metrics are, the better the performance is.

4.2 Ranking Performance.

We evaluate the ranking performance of our method SuRe compared to that of other methods.

Experimental setup. We perform this experiments on the Polblogs dataset. In Polbogs dataset, a node represents a blog, and an edge between nodes indicates a hyperlink between blogs. In the dataset, each node has a label which is either *liberal* or *conservative*. Among nodes connected from the query node (i.e., neighbors), we choose nodes having the same political position to the query node as positive nodes, and nodes having the opposite propensity to the query node as negative nodes. Note that the numbers of positive nodes and negative nodes do not exceed their query node's degree. We sample 115 nodes whose degrees are greater than 4 as query nodes to perform this experiment. We use all nodes except neighbors from a query node as test nodes. In this experiment, we aim to boost ranks of nodes having the same political position as the query node.

Case study. We analyze the ranking quality produced from each method in the Polblogs dataset. Table 4 shows the top-10 ranking list for a query node *obsidianwings*, a liberal blog. Red colored nodes are conservative, and the black colored ones are liberal. As shown in the table, our ranking result from SuRe is of a higher quality compared to those from RWR, SRW, and QUINT since top-10 ranking result from SuRe contains only liberal nodes while other ranking results have several conservative nodes, considering that the query node is liberal.

Result. To evaluate ranking performances, we measure MAP, Precision@20, and AUC for the ranking results produced from our method SuRe and other random walk based methods. For brevity, we report MAP and Precision@20 in Polblogs.

In Polblogs, if the query node is liberal (conservative), then the positive class is liberal (conservative), and the negative one is conservative (liberal). As shown in Figure 5, our method SuRe shows the best ranking performance compared to other methods in terms of MAP and Precision@20.

4.3 Link Prediction Performance.

We examine the link prediction performance of our proposed method SuRe compared to other link prediction methods as well as RWR-based methods SRW and QUINT.

Experimental Setup. In the link prediction task, we aim to predict future links from a query node to other nodes based on relevance scores. We perform this experiments on the HepPh and HepTh datasets which are time-stamped networks. Here, we focus on predicting links to nodes that are 2-hops away from the query node since most of new edges are created closing a triangle [5]. The experimental setting is as follows:

- We consider time-stamp information in networks to construct training and test datasets as in [5]. We select nodes whose degrees are greater than 30 as query nodes. For each query node s, let $t_{s,\min}$ and $t_{s,\max}$ denote the minimum and maximum time-stamp of 1-hop neighbors of node s, respectively. Suppose $t_{s,\min} < t_{s,1} < t_{s,2} < t_{s,\max}$. We select links ($s$, v) created between $t_{s,2}$ and $t_{s,\max}$ as test data where node v is 2-hop neighbors from node s before the links are created. We choose the query node's 1-hop neighbors created between $t_{s,1}$ and $t_{s,2}$ as positive nodes. We sample the same number of negative

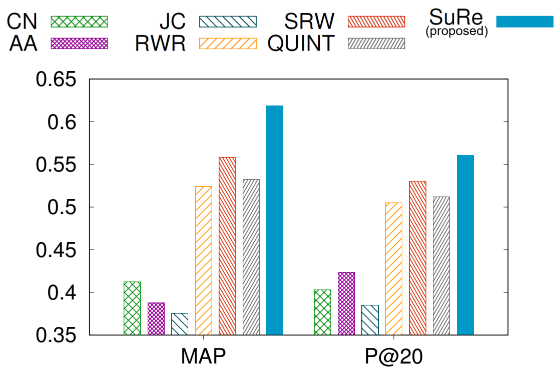


Fig. 6: Link prediction performance on the HepPh dataset. SuRe shows the highest accuracies: 10.8% higher MAP, and 5.7% higher Precision@20 compared to the best existing method.

Table 5: AUC result. We compare SuRe with other baselines including supervised methods SRW and QUINT for link prediction. SuRe provides the best link prediction accuracy.

Dataset	**SuRe**	RWR	SRW	QUINT
HepPh	**0.95516**	0.93597	0.94405	0.93612
HepTh	**0.96029**	0.94002	0.94845	0.94203

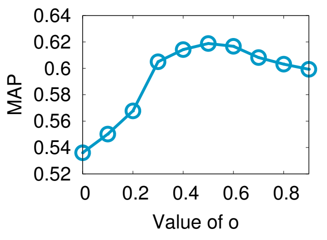
nodes as that of positive ones, which are 3-hops or more from node s. We exploit other links excepts the test links as training data. We select $t_{s,1}$ and $t_{s,2}$ such that $t_{s,1} = t_{s,\min} + 0.3L$ and $t_{s,2} = t_{s,\min} + 0.7L$, respectively, where $L = t_{s,\max} - t_{s,\min}$ is the total time length.

Result. Figures 1, 6, and Table 5 show the link prediction performances in terms of MAP, Precision@20, and AUC. As shown in the results, our method SuRe outperforms other competitors including SRW and QUINT which are the state-of-the-art methods for link prediction. In the HepTh dataset, compared to the best competitor SRW, SuRe achieves 14.7% improvement in terms of MAP, and 10.1% improvement in terms of Precision@20 (Figure 1). For the AUC results, SuRe gives the best performance as shown in Table 5. Note that SuRe provides the best prediction over all datasets. The results state that assigning a distinct restart probability to each node and learning the restart probabilities (RWER and SuRe) have a significant effect on link prediction compared to using a fixed restart probability for all nodes (RWR). Furthermore, the result indicates that learning restart probabilities (SuRe) provides better link prediction accuracy than existing supervised learning methods that focus on learning edge weights (SRW) or network topology (QUINT).

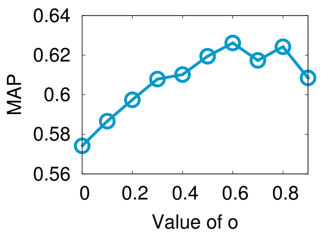
Table 6: Number of parameters for each supervised method.

Dataset	SRW [5]	**SuRe (proposed)**	QUINT [14]
$\#parameters$	$O(\#features)$	$O(n)$	$O(n^2)$

Discussion. We discuss the above experimental results in terms of the number of model parameters. As shown in Table 6, SRW has not enough parameters (i.e., $\#features < n$); thus, feature selection is important for the performance

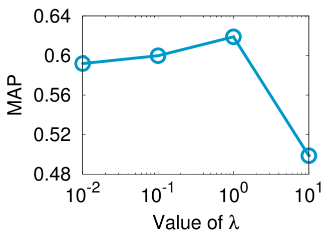


(a) MAP in the HepPh dataset

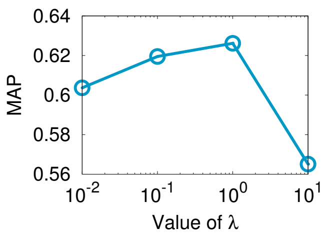


(b) MAP in the HepTh dataset

Fig. 7: Sensitivity of parameter **o** of our method SuRe in the HepPh and HepTh datasets. We report the link prediction accuracy using MAP measure, changing the values of the elements in the origin vector **o**, where all the elements of **o** are set to a same value. Note that the performance of SuRe is improved by introducing the origin parameter **o**, compared to not using **o** which corresponds to setting $\mathbf{o} = 0$.



(a) MAP in the HepPh dataset



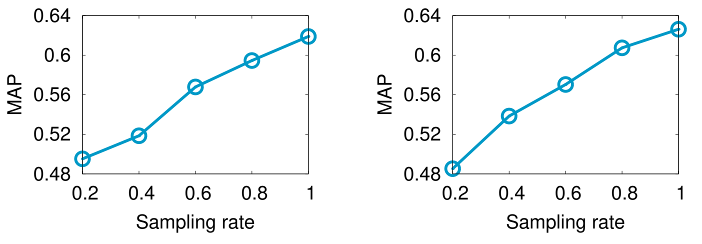
(b) MAP in the HepTh dataset

Fig. 8: Sensitivity of parameter λ of our method SuRe in the HepPh and HepTh datasets. We report the MAP scores in the link prediction task, varying the value of λ. Note that SuRe avoids overfitting problem and shows improvement by introducing the regularization parameter λ. When λ is $10^0 = 1$, SuRe exhibits the best link prediction performance in both datasets.

of applications in SRW. On the other hand, QUINT has too many parameters; thus, QUINT is prone to overfit. Also, it is infeasible to learn $O(n^2)$ parameters in large-scale graphs. Compared to these methods, SuRe has a moderate number of parameters, implying that 1) the expressiveness of SuRe is better than that of SRW, and 2) SuRe is not likely to overfit compared to QUINT. This point explains why SuRe provides better performance of the ranking and link prediction tasks than SRW and QUINT do as shown in Sections 4.2 and 4.3.

4.4 Parameter Sensitivity.

We investigate the parameter sensitivity of SuRe w.r.t. the value of the origin vector **o** and λ. The origin vector **o** serves as a model regularizer which helps avoid overfitting and improves accuracy, as described in Section 3.4. We evaluate MAP of the link prediction task, and report the results in Figure 7. Note that the performance of SuRe is improved by introducing the origin parameter **o**, compared to the case without **o**, which corresponds to the leftmost points in both plots of Figure 7.



(a) MAP in the HepPh dataset (b) MAP in the HepTh dataset

Fig. 9: Link prediction performance of SuRe varying the number of labeled nodes in the HepPh and HepTh datasets. We sample labeled nodes with a sampling rate. Note that a higher sampling rate leads to more training examples which in turn improve the performance.

λ is a regularization parameter that controls the importance of the regularization term. We evaluate MAP of the link prediction task in the HepPh and HepTh datasets varying the value of λ. As shown in Figure 8, when λ is 1, the MAP score of SuRe is the highest. By introducing the regularization parameter λ, SuRe avoids the overfitting problem and improves accuracy.

4.5 Effect of Number of Labeled Nodes

We evaluate the link prediction performance of SuRe by varying the number of labeled nodes. For a given seed node, we sample labeled nodes with a sampling rate. E.g., the average number of the labeled nodes in the Hepth dataset is 69.58; when the sampling rate is 0.2, then the average number of labeled nodes becomes 13.91 . As shown in Figure 9, when a seed node has many labeled nodes, SuRe shows better performance since it learns a better model with rich data.

4.6 Scalability.

We examine the scalability of our proposed method SuRe compared to other baselines. We perform SuRe to find the optimal restart probabilities and RWER to compute rankings with various sizes of the Wikipedia dataset to investigate the scalability of SuRe. For the dataset, we extract the principal sub-matrices, which are the upper left part of the adjacency matrix, of different lengths to get graphs of different number of edges. Figure 10 shows that SuRe scales near-linearly with the number of edges; this result is consistent with Theorem 3. The slope of the fitted line for SuRe is 0.76, the smallest number: those for RWR, SRW, and QUINT are 0.83, 0.88, and 2.57, respectively. Note that SuRe is the fastest among the supervised methods. Although RWR, which is an unsupervised method, is faster than SuRe, RWR shows low accuracy as shown in Figures 1, 5, and 6.

5 Related Works

The related works fall into two main categories: 1) relevance measures in graphs, and 2) ranking and link prediction based on relevance measures.

Relevance measures in graphs. There are various relevance measures in graphs based on link analysis and random walk, e.g., PageRank [17], HITS [12], Random Walk Graph Kernel [11], and RWR (or Personalized PageRank) [7]. Among these measures, RWR has received much attention from the data mining community since it provides a personalized ranking w.r.t. a node, and it has

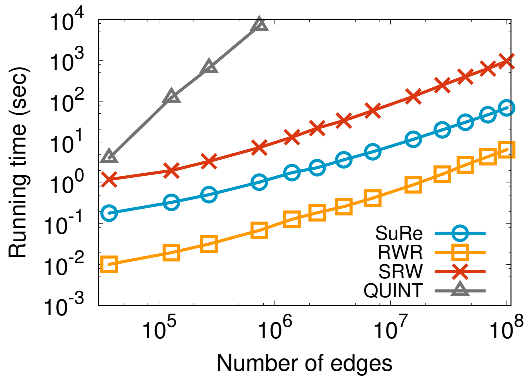


Fig. 10: Scalability of SuRe compared to baselines in the Wikipedia dataset. SuRe has the smallest slope 0.76 of the fitted line, while the slopes for RWR, SRW, and QUINT are 0.83, 0.88, and 2.57, respectively. Note that SuRe is the fastest among the supervised methods SRW and QUINT. Although SuRe shows slower running time compared to RWR which is an unsupervised method, SuRe provides higher accuracy than RWR in both ranking and link prediction tasks as shown in Figures 1, 5, and 6.

been applied to many graph mining applications such as community detection [4], link prediction [5, 14], ranking [27], and graph matching [26]. Also, fast and scalable methods [23, 10, 27] for computing RWR in large graphs have been proposed to boost the performance of those applications in terms of time.

Ranking and link prediction. Jung et al. [9] extended the concept of RWR to design a personalized ranking model in signed networks. Wang et al. [28] proposed an image annotation technique that generates candidate annotations and re-ranks them using RWR. Liben-Nowell et al. [15] extensively studied the link prediction problem in social networks based on relevance measures such as PageRank, RWR, and Adamic-Adar [1]. Many researchers have proposed supervised learning methods for link prediction. Backstrom et al. [5] proposed Supervised Random Walk (SRW), a supervised learning method for link prediction based on RWR. SRW learns parameters for adjusting edge weights. Li et al. [14] developed QUINT, a learning method for finding a query-specific optimal network. QUINT modifies the network topology including edge weights. In many real-world scenarios, however, modifying the graph structure would not be allowed. On the contrary, our SuRe method controls the behavior of the random surfer without modifying the graph structure, and provides better prediction accuracy than other competitors as shown in Section 4.

6 Conclusion

We propose Random Walk with Extended Restart (RWER), a novel relevance measure using distinct restart probabilities for each node. We also propose SuRe, a data-driven algorithm for learning restart probabilities of RWER. Experiments show that our method brings the best performance for ranking and link prediction tasks, outperforming the traditional RWR and recent supervised learning methods. Specifically, SuRe improves MAP by up to 14.7% on the best competitor. Future works include designing distributed algorithms for computing and learning RWER.

References

1. Adamic, L.A., Adar, E.: Friends and neighbors on the web. Social networks **25**(3), 211–230 (2003)
2. Agarwal, A., Chakrabarti, S.: Learning random walks to rank nodes in graphs. In: ICML (2007)
3. Agarwal, A., Chakrabarti, S., Aggarwal, S.: Learning to rank networked entities. In: KDD (2006)
4. Andersen, R., Chung, F., Lang, K.: Local graph partitioning using pagerank vectors. In: FOCS (2006)
5. Backstrom, L., Leskovec, J.: Supervised random walks: predicting and recommending links in social networks. In: WSDM (2011)
6. Fujimoto, T., Ranade, R.R.: Two characterizations of inverse-positive matrices: the hawkins-simon condition and the le chatelier-braun principle. Electronic Journal of Linear Algebra **11**(1), 6 (2004)
7. Haveliwala, T.H.: Topic-sensitive pagerank. In: WWW (2002)
8. He, J., Li, M., Zhang, H.J., Tong, H., Zhang, C.: Manifold-ranking based image retrieval. In: MM (2004)
9. Jung, J., Jin, W., Sael, L., Kang, U.: Personalized ranking in signed networks using signed random walk with restart. In: ICDM (2016)
10. Jung, J., Shin, K., Sael, L., Kang, U.: Random walk with restart on large graphs using block elimination. TODS **41**(2), 12 (2016)
11. Kang, U., Tong, H., Sun, J.: Fast random walk graph kernel. In: SDM, pp. 828–838. SIAM (2012)
12. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. JACM **46**(5), 604–632 (1999)
13. Langville, A.N., Meyer, C.D.: Google's PageRank and beyond: The science of search engine rankings. Princeton University Press (2011)
14. Li, L., Yao, Y., Tang, J., Fan, W., Tong, H.: Quint: On query-specific optimal networks. In: KDD (2016)
15. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American society for information science and technology **58**(7), 1019–1031 (2007)
16. Mozer, M.C.: Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic (2003)
17. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web. (1999)
18. Petersen, K.B., Pedersen, M.S., et al.: The matrix cookbook. Technical University of Denmark **7**, 15 (2008)
19. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., et al.: Numerical recipes, vol. 3. cambridge University Press, cambridge (1989)
20. Saad, Y., Schultz, M.H.: Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on scientific and statistical computing **7**(3), 856–869 (1986)
21. Salton, G., McGill, M.J.: Introduction to modern information retrieval (1986)
22. Sherman, J., Morrison, W.J.: Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. The Annals of Mathematical Statistics **21**(1), 124–127 (1950)
23. Shin, K., Jung, J., Lee, S., Kang, U.: Bear: Block elimination approach for random walk with restart on large graphs. In: SIGMOD (2015)
24. Sun, J., Qu, H., Chakrabarti, D., Faloutsos, C.: Neighborhood formation and anomaly detection in bipartite graphs. In: ICDM (2005)
25. Tong, H., Faloutsos, C.: Center-piece subgraphs: problem definition and fast solutions. In: KDD (2006)
26. Tong, H., Faloutsos, C., Gallagher, B., Eliassi-Rad, T.: Fast best-effort pattern matching in large attributed graphs. In: KDD (2007)
27. Tong, H., Faloutsos, C., Pan, J.Y.: Fast random walk with restart and its applications (2006)
28. Wang, C., Jing, F., Zhang, L., Zhang, H.J.: Image annotation refinement using random walk with restarts. In: MM (2006)
29. Zhu, Z.A., Lattanzi, S., Mirrokni, V.S.: A local algorithm for finding well-connected clusters. In: ICML (2013)