공학박사학위논문

실세계 그래프 특징을 활용한 랜덤 워크 기반 대규모 그래프 마이닝

Random Walk-based Large Graph Mining Exploiting Real-world Graph Properties

2020년 2월

서울대학교 대학원

컴퓨터공학부

정진홍

실세계 그래프 특징을 활용한 랜덤 워크 기반 대규모 그래프 마이닝

Random Walk-based Large Graph Mining Exploiting Real-world Graph Properties

지도교수 강 유

이 논문을 공학박사 학위논문으로 제출함

2020년 1월

서울대학교 대학원

컴퓨터공학부

정진홍

정진홍의 박사 학위논문을 인준함

2019년 12월

위 원 장	문봉기	(인)
부 위 원 장	강 유	(인)
위 원	김형주	(인)
위 원	김상욱	(인)
위 원	이영기	(인)

Abstract

Random Walk-based Large Graph Mining Exploiting Real-world Graph Properties

Jinhong Jung
Department of Computer Science & Engineering
College of Engineering
The Graduate School
Seoul National University

Numerous real-world relationships are represented as graphs such as social networks, hyperlink networks, and protein interaction networks. Analyzing those networks is important to understand the real-life phenomena. Among various graph analysis techniques, random walk has been widely used in many applications with satisfactory results. However, various real-world graphs are large and complicated with diverse labels. Traditional random walk based methods require heavy computational cost, and disregards those labels for performing random walks; thus, its utilization has been limited in such large and complicated graphs.

In this thesis, I handle the technical challenges of mining large real-world graphs based on random walk. Real-world graphs have distinct structural properties which become a basis to increase the performance of the random walk in terms of speed and quality. Based upon this idea, I develop fast, scalable, and exact methods for node

ranking using random walk in large-scale plain networks. I also design accurate models using random walks for node ranking and relational reasoning in labeled graphs such as signed networks and knowledge bases.

Through extensive experiments on various real-world graphs, I demonstrate the effectiveness of the methods and models proposed by this thesis. The proposed methods process 100× larger graphs, and require up to 130× less memory with up to 9× faster speed compared to other existing methods, successfully scaling to billion-scale graphs. Also, the proposed models substantially improve the predictive performance of a variety of tasks in labeled graphs such as signed networks and knowledge bases.

Keywords : Graph Mining, Random Walk in Graphs, Random Walk with Restart Models, Real-world Graph Properties, Large-scale Graphs, Signed Networks, Edge-Labeled Graphs

Student Number : 2015-31053

Contents

List of Figures

List of Tables

Chapter 1

Overview

1.1 Motivation

Graphs are fundamental data structures modeling any relationships between entities. Each entity is abstractly represented as a *node*, and each of the related pair of nodes is symbolized as an *edge*. Numerous real-world phenomena around us are naturally modeled by graphs (or networks) [1]. For example, social networks from online social services such as Twitter [2] and Facebook [3] have been widely used to represent friendships of people. Hyperlink networks [4] express connections between pages on linked knowledge systems such as the Web or Wikipedia. Citation networks connect a scholarly paper to other papers in its bibliography [5]. In bioinfometics, protein networks [6] are used to represent various interactions between proteins. The human brain is also represented as a network of nerve cells [7]. Nature creates food web networks where species are connected by links indicating which species feed on which other species [8].

Graph mining has attracted considerable attention from diverse research fields since it enables us to gain a better understanding of the complicated relationships in the real world through analyzing their networks. On top of the insight into the graphs, many researchers have developed beneficial applications and improved the performance of their specific tasks based on graphs. For instance, social network analysis has revealed plenty of interesting knowledge on social events and human behav-

ior [9, 10, 11]. It has further led to the emergence of network science [12] which provides a solid theoretical background to understand various real-world networks. As another example, web search engines such as Google [13] have improved their search performance by utilizing hyperlink network analysis which reveals node importance as a ranking score in graphs.

Random walk [14, 15] has been extensively studied and utilized as a simple but powerful tool for looking into graph data. This technique aims to simulate a user who randomly moves around nodes in a graph with a specific purpose (e.g., a web surfer jumps from page to page on the Web in order to search target information), thereby resulting in relevance or ranking scores between nodes. There are several models on random walk in graphs such as HITS [16], SimRank [17], PageRank [13], Random walk with Restart (RWR) [18], etc. Among those models, RWR has been popular in academic fields as well as industrial areas because it is able to capture node-to-node relevance scores personalized to a query node (due to this point, RWR is called Personalized PageRank). Moreover, many works [19, 20] have empirically shown that RWR has a good ability to account for the multi-faceted relationships (e.g., multiple connections, path lengths, node degrees, etc) between nodes and consider the global topology of a network at the same time. Thus, RWR effectively obtains personalized node relevance scores in graphs, and it has been extensively used for a variety of applications. Table 1.1 lists well-known graph mining applications in which RWR and its variants have been frequently used.

Even though the importance of such random walk models including RWR is being emphasized, the utilization of those models is limited for analyzing further large and complicated real-world networks due to several technical obstacles. The main limitation of the random walk based techniques is that they require enormous com-

Table 1.1: **Various applications based on random walk techniques**

Application	Brief Description	References
Node ranking	Rank nodes such as web pages in order of a specific importance in a graph	[21, 13, 22] [23, 24, 25]
Link prediction	Predict future links to be connected between nodes in a graph	[26, 27, 28] [29, 30, 31]
Recommendation	Suggest interesting items such as movie and music to a specific user in a user-item network	[19, 18, 32] [33, 34, 35]
Subgraph mining	Extract meaningful subgraphs between two or more nodes	[36, 20, 37] [38, 39, 40]
Anomaly detection	Identify rare items, events or observations such as web spams, anomalies, and bank frauds	[41, 42, 43] [44, 45, 46]
Community detection	Detect groups of nodes having similar affiliations different to the rest of a graph	[47, 48, 49] [50, 51, 52]

putational cost, especially in large graphs. Recent stunning advances in computing and networking technologies have led to graphs of unprecedented size. For instance, Wikipedia described that it comprises more than 40 millions articles in 2015[1]. Facebook reported that it had about 2.41 billion monthly active users in 2019[2]. As a result, traditional methods fail to perform the random walk based analyses for very large graphs in a reasonable time with restricted resources. Many researchers have made great efforts to tackle this efficiency issue in large-scale graphs, and proposed various types of methods such as exact [53, 13, 54], approximate [55, 56, 57, 58], and top-k approaches [59, 60, 61, 62]. However, none of such previous methods satisfy all of the desirable aspects such as speed, scalability, exactness, and versatility[3] when it comes to the random walk computation in large graphs.

In addition to the computational problem, the traditional random walk models

[1] https://en.wikipedia.org/wiki/Wikipedia

[2] https://newsroom.fb.com/company-info

[3] The top-k approach focuses on finding top-k relevant nodes under the random walk mechanism. They emphasize the efficiency, but their versatility is limited because they cannot be applied to many graph mining applications [48, 27, 50, 19, 26, 41, 37, 49] requiring the random walk scores of all nodes.

have another limitation that no labels on nodes and edges are allowed in graphs. The main reason is that the traditional random surfer does not consider such label information when doing random walk. Due to this point, the classic models had to ignore the label data although complicated relationships between nodes are modeled by various labels in many real-world networks. For example, signed networks [63] have been suggested to model trust relationships between people with positive and negative edges. Knowledge bases [64] represent diverse predicates between subjects and objects as edge labels. The problem is that this although label information plays a key role in distinguishing unique characteristics of such networks, the traditional models do not take into account labels at all. Thus, the use of the traditional random walk and its quality are limited in such labeled networks. Although several random walk based variants [65, 66, 67] have been proposed to utilize this label information, most of them are based on heuristic techniques, and exhibit the unsatisfactory performance for applications in those labeled networks.

1.2 Research Statement

This section describes the research statement which summarizes the research goals, importance, technical challenges, main approaches, and contributions of this thesis.

1.2.1 Research Goals and Importance

I aim at *devising fast, scalable, and exact methods and designing effective models for random walk based mining on large real-world graphs* through this research.

First of all, I focus on achieving all of speed, scalability, and exactness when computing the random walk scores of RWR in very large graphs having billions of edges on a single machine. It is not easy to handle the problem of fulfilling all of the

computational aspects in such billion-scale graphs where most previous methods had to sacrifice at least one of them. For example, approximate approaches lose accuracy to boost efficiency, and exact preprocessing methods suffer from a scalability issue in large graphs due to the guarantee of exactness. How can we compute the RWR scores quickly and exactly in billion-scale graphs, especially without borrowing the power of multiple machines? What are other existing methods for RWR missing in terms of such computational factors for processing large graphs?

Further, I address how to utilize edge labels with random walk in real-world networks. As described in Section 1.1, the label information is crucial as it represents the nature of such labeled networks; thus, it should have been considered when doing random walk in graphs. However, previous research works are insufficient for providing a clear solution for the problem, i.e., our understanding of random walk with such labels was nascent. How can we utilize various labels involved in complicated real-world networks with random walk for effective graph mining? What does the label information mean for random walk in such labeled graphs?

Through this thesis, I concentrate on finding fundamental solutions of the above questions to pursue the research goal. It is significantly important to accomplish this goal since the methods derived from this research will enable researchers and practitioners to efficiently analyze large real-world graphs that previous approaches could not process within restricted resources. This research will also lead to allowing developers to build their novel, beneficial, and high-quality applications which effectively utilize label data contained in real-world graphs. Furthermore, I aim at establishing theoretical backgrounds on the approaches taken by this thesis, and hope these to pave the way for future research on random walk in graphs and graph mining.

1.2.2 Technical Challenges

The technical challenges that this thesis needs to address are categorized as follows.

Computational Performance Improvement. The main challenge is how to reduce the enormous cost incurred by the RWR computation in large-scale graphs. More specifically, it is extremely challenging to balance between speed, scalability, and exactness for computing RWR scores if a graph has billions of nodes and edges. The RWR model aims to obtain the random walk scores of all nodes w.r.t. a given query node in a graph (it is often called *single-source RWR* [68]). If a user gives a different query node, a method needs to repeat the RWR computation for the given query node where the computational burden for each query node cannot be ignored. Also, it is infeasible to store all precomputed scores for each query node, especially in billion-scale graphs, since this approach requires $O(n^2)$ space where n is the number of nodes. Thus, a desirable method should compute the RWR scores quickly using less memory usage without loss of accuracy whenever a query node is given.

To avoid such tremendous cost, many researchers have exploited approximate or top-k approaches. However, the accuracy sacrificed by the approximate methods [55, 56, 57, 58] is not sufficient considering their computational improvement, and significantly degrades the quality of applications using RWR. The top-k approach [59, 60, 61, 62] focuses on finding top-k relevant nodes under the RWR model; thus, it is limited to use this approach in many graph mining applications [48, 27, 50, 19, 26, 41, 37, 49] requiring all nodes' scores. There are also noticeable works [69, 70, 71, 72] to increase the scalability of the RWR computation with the guarantee of exactness borrowing the power of distributed systems. However, most distributed methods have focused on fitting existing algorithms to distributed systems, i.e., they optimized the I/O costs of power iteration, one of traditional iterative methods for RWR, so that a distributed

system scales to large graphs based on the algorithm. Hence, in this thesis, one primary goal is to devise novel algorithms for faster and more scalable computation for exact RWR scores under a single machine so that our algorithms can be extended to distributed machines as future work.

Label Data Utilization. Another challenge is how to build random walk models which effectively exploits the label data involved in real-world graphs. Many relationships between nodes are represented as labels on edges where such networks are called *edge-labeled graphs*. In this thesis, categorical labels are considered since such categorical labels can be interpreted as a specific sense in the input network's domain. For instance, signed networks allow an edge to have a positive or negative label (or sign) as trust relationship. Knowledge bases let an edge have a verb connecting two entities where the verb is represented as one of multiple labels. Although the random walk techniques including RWR well capture relevance scores between nodes in plain graphs, it is not obvious how the label data should be reflected into random walks in such edge-labeled graphs. In this thesis, I will first deliberate how the labels should be interpreted when its comes to node relevance scores measured by random walk models in edge-labeled graphs. Based on this concept, I will design novel models providing a solution on how a random surfer should treat such labels during its random walk.

1.2.3 Main Approaches

I describe the main approaches of this thesis to tackle the aforementioned technical challenges.

Exploiting Real-world Graph Properties. One of the main approaches used in this thesis is to exploit distinct properties inherent in real-world graphs. Most pre-

vious methods have focused on improving the computational performance and the quality of the random walk without carefully investigating graph data. As a result, those existing methods have limitations in dramatically improving the performance of random walk, especially in large-scale and complicated graphs. To make a breakthrough, I notice most real-world graphs have distinct structural properties commonly appeared according to their domain. For example, real-world graphs are scale-free, i.e., their degree distributions are highly skewed following a power-law [73]. In real-world signed networks, certain types of signed triangles dominate those networks, i.e, balanced triangles are more likely to be created than unbalanced ones [74, 75] (see details in Chapter 4). Our previous works [54, 76] have shown that such scale-free property is helpful for reducing computational cost of operations based on adjacency matrices of real-world graphs. Thus, I utilize various unique and structural properties of real-world graphs to form a basis of solutions for boosting the performance in terms of speed and quality of random walk.

Numerical Computation Methods. Most computations about random walk in graphs are mathematically represented by operations related to adjacency matrices of the graphs. Thus, it is important to understand and utilize efficient numerical computation methods in order to accelerate the speed on the computations on random walk, especially when we use a single machine. To efficiently process large-scale graphs, I combine the approach exploiting real-world graph properties and various numerical computation methods from simple operations such as sparse matrix-vector multiplication to advanced operations such as preconditioned Krylov iterative linear solver [77, 78, 79].

Linear Algebra and Stochastic Process. In order to make random walk based models computable, mathematical theories such as linear algebra and stochastic pro-

cess are necessary because operations on graphs lead to matrix computations, and the behavior of random walks is described by a Markov chain in stochastic process (i.e., it guarantees the random walk distribution becomes stable as the number of random walks goes to infinity). In this thesis, I aim to design new random walks based models exploiting edge labels; thus, such theories are used to mathematically formalize the concepts and equations of the novel models and guarantee their convergence behavior in labeled real-world graphs.

1.2.4 Contributions

I provide a brief summary of the main contributions resulted from this dissertation as follows:

- **Fast and Scalable Ranking in Large-scale Plain Graphs (Chapter 3).** First, I propose BePI for random walk based ranking on billion-scale graphs by exploiting real-world graphs structures and taking the advantage of both preprocessing and iterative approaches. BePI processes 100× larger graphs, and requires up to 130× less memory space than other preprocessing methods. In the query phase, BePI computes node ranking scores up to 9× faster than existing methods.
- **Personalized Ranking in Signed Networks (Chapter 4).** Second, I propose SRWR for personalized ranking in signed networks by introducing a signed random surfer with balance theory describing signed triangle patterns in real-world signed networks. SRWR achieves the best accuracy for link prediction, predicts trolls 4× more accurately, and shows a satisfactory performance for inferring missing signs of edges compared to other competitors. I also develop SRWR-Pre for fast ranking in signed networks by exploiting real-world graph

structures in a preprocessing manner. SRWR-Pre preprocesses a signed network 4.5× faster and requires 11× less memory space than other preprocessing methods; furthermore, SRWR-Pre computes SRWR scores up to 14× faster than other methods in the query phase.

- **Relational Reasoning in Edge-labeled Graphs (Chapter 5).** Finally, I design a random walk based model MuRWR for accurate relation reasoning in edge-labeled graphs. I introduce a labeled random surfer, and learn appropriate rules on changing the surfer's label from an input graphs for effective reasoning. MuRWR provides the most accurate performance for relation inference in diverse edge-labeled graphs compared to its competitors.

1.2.5 Overall Impact

My research outcomes have potential impacts on academia and industrial worlds as the followings:

- **Computational Improvement.** My approaches significantly improve the computational performance of various graph mining tasks based on random walk in terms of speed, space, and scalability.
- **Effective Analysis.** My novel models are beneficial for researchers to effectively analyze complex graphs such as signed networks and knowledge base, inducing various applications.

I made most of the algorithms developed throughout this thesis open to the public for reproducibility and the benefit of the community. In addition, this research achieved the following results:

- The work [80] was awarded the silver prize of Samsung Humantech Paper Award, one of the most prestigious paper awards in Korea.

- The researches [80, 22, 76, 81] were supported by Global Ph.D. Fellowship Program of National Research Foundation and NAVER Ph.D. Fellowship.
- This research results in 2 domestic patent applications and 2 registered domestic patents related to the proposed methods [80, 22, 81].

1.3 Thesis Organization

The rest of this thesis proposal is organized as follows. The background on several concepts and the survey of previous works related to this thesis are provided in Chapter 2. In Chapter 3, I present the fast and scalable algorithm BePI for random walk based ranking in large-scale graphs. In Chapter 4, I first describe the novel model SRWR for personalized ranking and its iterative method, and show the fast method for computing SRWR in signed social networks. In Chapter 5, I propose our new model MuRWR based on random walk for relational reasoning in edge-labeled graphs. After discussing future work in Chapter 6, I conclude in Chapter 7.

Chapter 2

Background

In this chapter, I introduce concepts and notations on graphs and Random Walk with Restart (RWR), which are used throughout this dissertation, and describe previous works related to this research.

2.1 Definitions

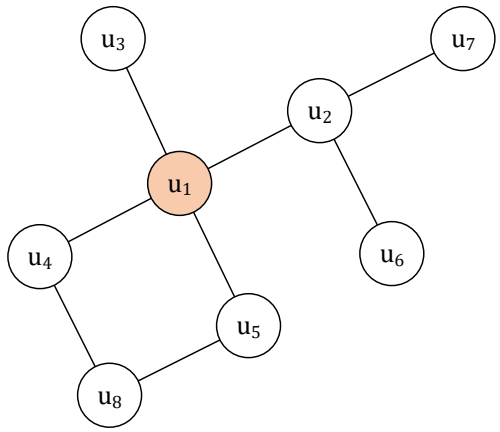
This section describes the definition and mathematical notations for graphs and RWR.

2.1.1 Notations on Graphs

I describe the definitions of graphs used throughout this thesis. Definition 2.1 defines a plain graph without any labels, which is mainly used in Chapter 3. Definition 2.2 describes signed networks having two edge labels, which is handled in Chapter 4 Edge-labeled networks having K edge labels is defined in Definition 2.3 which is used in Chapter 5.

Definition 2.1 (Plain Graph and Adjacency Matrix)**.** *A plain graph $G = (\mathbf{V}, \mathbf{E})$ is a pair of the two set $\mathbf{V}$ of nodes and the set $\mathbf{E}$ of edges where an edge (u, v) represents a connection between nodes u and v. The adjacency matrix $\mathbf{A}$ of G is a sparse matrix such that $\mathbf{A}_{uv}$ is 1 if there is an edge from u to v, and 0 otherwise.* ■

Definition 2.2 (Signed Graph and Signed Adjacency Matrix)**.** *A signed graph $G = (\mathbf{V}, \mathbf{E}^{\pm})$ consists of $\mathbf{V}$ of nodes and the set $\mathbf{E}^{\pm}$ of signed edges. A signed edge (u, v) is associated with a positive or negative sign, i.e., $(u, v, +)$ or $(u, v, -)$, respectively. The*



node	score	rank
u_1	0.287	1
u_2	0.171	2
u_3	0.068	6
u_4	0.124	3
u_5	0.124	4
u_6	0.054	7
u_7	0.054	8
u_8	0.118	5

Figure 2.1: **Example of RWR.** In the example, the query node is u_1 and RWR scores w.r.t. u_1 are presented in the table. The RWR scores are utilized for personalized ranking or link recommendation for u_1.

signed adjacency matrix $\mathbf{A}$ *of* G *is a sparse matrix such that* $\mathbf{A}_{uv}$ *is* $+1$ *(or* -1*) if there is a positive (or negative) edge* (u, v)*, and* 0 *otherwise.* ■

Definition 2.3 (Edge-labeled Graph and Labeled Adjacency Matrix)**.** *An edge-labeled graph* $G = (\mathbf{V}, \mathbf{E}, \mathbf{L})$ *consists of the set* $\mathbf{V}$ *of nodes, the set* $\mathbf{E}$ *of directed edges, and the set* $\mathbf{L}$ *of edge labels. Let* $\mathbf{L} = \{l_1, \cdots, l_K\}$ *where* l_k *is* k*-th label, and* K *is the number of edge labels. For each edge* $u \to v \in \mathbf{E}$ *such that* $u, v \in \mathbf{V}$*, the edge is associated with an edge label* $l_{uv} \in \mathbf{L}$*. The labeled adjacency matrix* $\mathbf{A}$ *of* G *is a sparse matrix such that* $\mathbf{A}_{uv}$ *is* l_{uv} *if there is an* l_{uv}*-labeled edge from* u *to* v*, and* 0 *otherwise.* ■

2.1.2 Random Walk with Restart

Given a graph G, a query node s, and a restart probability c, random walk with restart (RWR) [82] measures proximity scores $\mathbf{r}$ between the query node s and each node on the graph. RWR leverages the proximities by allowing a random surfer to move around the graph. Suppose that a random surfer starts at node s, and takes one of the following actions at each node:

- **Random Walk.** The surfer randomly moves to one of the neighbors from the current node with probability $1 - c$.
- **Restart.** The surfer goes back to the query node s with probability c.

The proximity or the RWR score between a node u and the query node s is the steady-state probability that the surfer is at node u after performing RWR starting from node s. If the proximity is high, we consider that nodes u and s are highly related, e.g., they are close friends in a social network. Thus, RWR provides relevance scores between the query node s and each node, and it is utilized as a personalized ranking for the query node s [82].

For example, suppose u_1 is the query node as shown in Figure 2.1. The RWR scores w.r.t. u_1 are presented in the table of the figure, and the scores are used for the personalized ranking for u_1. Also, we are able to recommend to friends for u_1 based on the scores. The RWR score of u_8 is higher than that of u_6 because u_8 is highly correlated to u_1 by the connections with u_4 and u_5. Thus, u_8 will be recommended to u_1 rather than u_6 would based on the RWR scores.

RWR scores for all nodes w.r.t. the query node s are represented as an RWR score vector $\mathbf{r}$ which is defined by the following recursive equation [13, 82] :

$$\mathbf{r} = (1 - c)\tilde{\mathbf{A}}^{\mathbf{T}}\mathbf{r} + c\mathbf{q} \tag{2.1}$$

where $\tilde{\mathbf{A}}$ is the row-normalized adjacency matrix of the graph G, and $\mathbf{q}$ is the starting vector whose entry that corresponds to the node s is set to 1, and others to 0. From Equation (2.1), we obtain the following linear equation:

$$(\mathbf{I} - (1 - c)\tilde{\mathbf{A}}^{\mathbf{T}})\mathbf{r} = c\mathbf{q} \Leftrightarrow \mathbf{H}\mathbf{r} = c\mathbf{q} \tag{2.2}$$

where $\mathbf{H} = \mathbf{I} - (1-c)\tilde{\mathbf{A}}^{\mathbf{T}}$. Note that $\mathbf{q}$ is an RWR query, and $\mathbf{r}$ is the result corresponding to the query. $\mathbf{q}$ is determined by the query node s, and $\mathbf{r}$ is distinct for each RWR query. RWR is a special case of Topic-specific PageRank which sets multiple seed nodes in the starting vector $\mathbf{q}$ while RWR sets only one seed node [13, 21].

2.2 Related Works

In this section, I survey existing research works related to random walk-based graph mining.

2.2.1 Previous Methods for RWR in Plain Graphs

I review previous works on the RWR computation from two perspectives: 1) iterative methods, 2) preprocessing methods, and 3) approximate and top-k methods. The iterative and preprocessing methods focus on computing exact single-source RWR scores of all nodes w.r.t. a given query node. The approximate methods compute approximate single-source RWR scores, and top-k methods aim at finding top-k relevance nodes in order of RWR scores approximately or exactly.

Iterative methods for RWR. The most well-known method is power iteration [13] which repeatedly updates the RWR score vector in Equation (2.1). The power iteration method aims at finding the eigenvector corresponding to the largest eigenvalue of the Google matrix derived from Equation (2.1). Krylov subspace methods [79] are also used to compute the solution of a linear system shown in Equation (2.2). These methods iterate a procedure to search the solution in the Krylov subspace. Since the matrix $\mathbf{H}$ is non-singular and non-symmetric [83], any Krylov subspace method, such as GMRES [53], which handles a non-symmetric matrix, can be applied to Equation (2.2). While these iterative methods do not require preprocessing, they

have expensive query cost especially when there are lots of queries, since the whole iterations need to be repeated for each query.

Preprocessing methods for RWR. The query speed of RWR can be accelerated significantly by precomputing the inverse of $\mathbf{H}$ of Equation (2.2). However, matrix inversion does not scale up for large graphs, as it involves a dense matrix that is too large to fit in memory. To tackle this problem, alternative preprocessing methods have been developed. Tong et al. [82] proposed NB_LIN, which decomposes the adjacency matrix using a low-rank approximation in the preprocessing phase, and approximates $\mathbf{H}^{-1}$ from the decomposed matrices in the query phase. Fujiwara et al. applied LU decomposition [60] and QR decomposition [84] to the adjacency matrix to obtain sparser matrices to use in place of $\mathbf{H^{-1}}$. Prior to applying LU decomposition [60], they reordered $\mathbf{H}$ based on the degree of nodes and the community structure to make $\mathbf{L^{-1}}$ and $\mathbf{U^{-1}}$ sparse. Bear [54, 76] preprocesses the adjacency matrix by exploiting node reordering and block elimination techniques. While all of these methods made performance improvements over previous approaches, they suffer from the scalability problem when it comes to billion-scale graphs.

Approximate and top-k methods for RWR. Iterative and preprocessing approaches often fail to scale up for real-world applications due to high computational cost. Several approximate methods have been developed to overcome this problem. Observing that the relevance scores are highly skewed, and real-world graphs often exhibit a block-wise structure, Sun et al. [41] proposed an approximate algorithm that performs RWR only on the partition containing the seed node, while setting the relevance score of other nodes outside the partition to 0. Building on similar observations, Tong et al. [82] proposed approximate algorithms, B_LIN and its derivatives, in which they applied a low-rank approximation to the cross-partition links using eigenvalue

decomposition. Gleich et al. [55] proposed methods that apply RWR only to a part of the graph, which is determined adaptively in the query phase. Andersen et al. [48] presented an algorithm for local graph partitioning problem that computes PageRank vectors approximately. Fast-PPR, a Monte Carlo-based method proposed by Lofgren et al. [57], estimates the single pair PPR (Personalized PageRank) between a start node and a target node by employing a bi-directional scheme. Bahmani et al. [85] developed a fast MapReduce algorithm based on Monte Carlo simulation for approximating PPR scores. To compute PPR approximately, Xie et al. [58] used a model reduction approach where solutions are projected to a low dimensional space. Also, several works have been proposed to focus on the k most relevant nodes w.r.t. a seed node instead of calculating the RWR scores of every node. K-dash, a top-k method proposed by Fujiwara et al. [60], computes the RWR scores of top-k nodes by exploiting precomputed sparse matrices and pruning strategies. Wu et al. [61] proposed Fast Local Search (FLOS) which finds top-k relevant nodes in terms of various measures including RWR. However, approximate and top-k computation for RWR scores are insufficient for many data mining applications [86, 48, 18, 27, 87, 82, 88] which require accurate RWR scores for any pair of nodes.

2.2.2 Ranking Models in Signed Networks

I review related works on random walk and ranking models in signed networks, which are categorized as follows: 1) ranking in signed networks, and 2) applications of ranking in signed networks.

Ranking in signed networks. Many researchers have made great efforts to design global node rankings in signed networks. Kunegis et al. [63] presented Signed spectral Ranking (SR) that heuristically computes PageRank scores based on a signed

adjacency matrix. Wu et al. [66] proposed Troll-Trust model (TR-TR) which is a variant of PageRank. In the algorithm, the trustworthiness of an individual user is modeled as a probability that represents the underlying ranking values. Shahriari et al. [65] suggested Modified PageRank (MPR), which computes PageRank in a positive subgraph and a negative subgraph separately, and subtracts negative PageRank scores from positive ones. Although the idea of MPR is easily applicable to other personalized ranking models such as RWR by computing ranking scores on the positive and negative subgraphs, this results in many disconnections between nodes. Note that all those models mainly focus on global node rankings, and they do not consider complex relationships between negative and positive edges such as friend-of-enemy or enemy-of-friend.

Applications of ranking in signed networks. Many applications in signed social networks such as link prediction, troll detection, and sign prediction have been studied in many literatures. Song et al. [89] proposed GAUC (Generalized AUC) to measure the quality of link prediction in signed networks where the link prediction task is to predict nodes which will be positively or negatively linked by a node in the future. They devised a matrix factorization based method GAUC-OPT which approximately maximizes GAUC for link prediction. Kunegis et al. [63] analyzed the Slashdot dataset from the perspective of troll detection, and proposed Negative Rank (NR) as a variant of PageRank for detecting trolls who behave abnormally in the social network. Leskovec et al. [90] proposed LOGIT which is specially designed for sign prediction classifying the sign between two arbitrary nodes. They exploited a logistic classifier trained by node and edge features such as node degrees and common neighbors between those two nodes. Guha et al. [91] also studied sign prediction, and devised TRUST measuring trustworthiness between two source and target nodes

by propagating trust and distrust from the source node to the target node.

2.2.3 Relational Reasoning Models in Edge-labeled Graphs

I review related works on random walk and relational reasoning in edge-labeled graphs as follows: 1) random walk on heterogeneous networks, and 2) relation inference in edge-labeled graphs.

Random walk on heterogeneous networks. RWR has received much attention and has been applied to many graph mining tasks. However, RWR has a limitation on predicting the relation between two nodes in edge-labeled graphs since it does not consider edge labels for its relevance. Several techniques [23, 33] have been proposed to compute RWR in heterogeneous networks. These methods focus on how to determine the weights of edges by exploiting attributes in the networks, and then construct a transition matrix with the weights to compute RWR. However, they also cannot infer the relation between the nodes in edge-labeled graphs because they produce only one relevance score between two nodes, similarly to RWR. For relation inference, we need to obtain K relevance scores for edge labels between two nodes if a graph has K edge labels. Many researchers have recently made great efforts to apply RWR for relevance between nodes in signed networks, a special type of edge-labeled graphs, represented by positive (trust) and negative (distrust) edges. Modified RWR (MRWR) [65] computes RWR as trust and distrust scores in positive and negative subgraphs, respectively. Although the idea is applicable to edge-labeled graphs by computing RWR on each subgraph containing only a specific edge label, this leads to many disconnections between nodes; thus, MRWR is unable to exploit meaningful patterns from multi-hop paths.

Relation inference in edge-labeled graphs. Two major approaches on re-

lation inference for multiple edge labels are classified into *path feature model* and *translation based model*. Path Ranking Algorithm (PRA) [67] is commonly used as a path feature model in heterogeneous networks. PRA extracts paths connecting two nodes, and exploits a random surfer to measure path probabilities which are used as features when predicting their relation. PRA, however, requires explicit path enumeration which becomes computationally problematic when it comes to long paths. Although the authors presented heuristic pruning techniques, PRA's inference has still been restricted to short paths since the path enumeration essentially has an exponential complexity to path length. Translation based models such as TransE [92] and TransR [93] have been widely utilized due to its simplicity and effectiveness for modeling relational data. They formulate the relation between two nodes as a translation between the corresponding node embeddings. However, those models consider only one directed edge at a time in training; hence, their reasoning is likely to miss the information provided by multi-hop paths between them.

Chapter 3

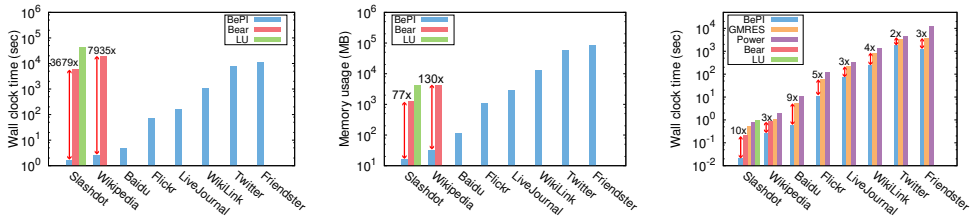
Fast and Scalable Ranking in Large-scale Plain Graphs

3.1 Introduction

Identifying node-to-node proximity in a graph is a fundamental tool for various graph mining applications, and has been recognized as an important research problem in the data mining community [94, 95, 60, 61]. Random walk with restart (RWR) provides a good relevance score, taking into account the global network structure [19] and the multi-faceted relationship between nodes [20] in a graph. RWR has been successfully utilized in many graph mining tasks including ranking [82], recommendation [96], link prediction [27], question and answering [97], and community detection [48, 50, 49].

Existing methods for scalable computation of RWR scores can be classified into two categories: iterative approaches and preprocessing approaches. Iterative methods, such as power iteration [13], compute an RWR score by repeatedly updating it until convergence. While they require much less memory space compared to preprocessing methods, they are slow in the query phase because matrix-vector multiplications should be performed each time for a different query node. This makes iterative methods not fast enough for billion-scale graphs.

On the other hand, preprocessing methods compute RWR scores using precomputed intermediate matrices. Since preprocessed matrices need to be computed just



(a) Preprocessing time (b) Memory space for preprocessed data (c) Query time

Figure 3.1: **Performance of BePI.** (a) and (b) compare the preprocessing time, and the memory space for preprocessed data, respectively, among preprocessing methods; (c) compares the query time among all tested methods. Bars are omitted if the corresponding experiments run out of memory or time (more than 24 hours). (a) In the preprocessing phase, BePI is the fastest and the most scalable among all preprocessing methods. Only BePI successfully preprocesses billion-scale graphs such as Twitter and Friendster datasets. (b) BePI uses the least amount of space for preprocessed data across all the datasets. Only BePI preprocesses all the datasets, whereas Bear and LU decomposition fail except for the two smallest ones. (c) In the query phase, BePI computes RWR scores faster than other competitors over all datasets. Details on these experiments are presented in Section 3.5.

once, and then can be reused, they are fast in the query phase, especially when they should serve many query nodes. However, existing preprocessing approaches have high memory requirements in common, due to the space for the preprocessed matrices, which makes it difficult to scale them up to billion-scale real-world graphs such as Twitter or Friendster datasets (see Table 3.2).

In this work, we propose BePI (Best of Preprocessing and Iteartive approaches for RWR), a fast, memory-efficient, and scalable method for computing RWR on billion-scale graphs. BePI addresses the challenges faced by previous approaches by combining the best of both preprocessing and iterative methods. BePI uses a block elimination approach, which is a preprocessing method, to achieve fast query time. BePI incorporates an iterative method within the block elimination to decrease memory requirements by avoiding expensive matrix inversion. The performance of BePI is further enhanced via matrix sparsification and preconditioning. Through extensive experiments with various real-world graphs, we demonstrate the superiority of BePI

over existing methods as shown in Figure 3.1. The main contributions of this paper are the followings:

- **Algorithm.** We propose BePI, a fast, memory-efficient, and scalable algorithm for computing RWR on billion-scale graphs. BePI efficiently computes RWR scores based on precomputed matrices by exploiting an iterative method, reducing the number of non-zeros of a matrix, and applying a preconditioner.
- **Analysis.** We give theoretical guarantees of the accuracy of BePI. We also analyze the time and the space complexities of BePI, and show that the complexities are smaller than those of the state-of-the art method.
- **Experiment.** BePI processes 100× larger graphs and requires 130× less memory space than existing preprocessing methods. Moreover, BePI provides near linear scalability in terms of preprocessing and query cost. BePI computes RWR scores up to 9× faster than existing iterative methods.

The code of our method BePI and datasets used in the paper are available at **https://datalab.snu.ac.kr/bepi**. The rest of the paper is organized as follows. In Section 3.2, we give preliminaries on the definition and algorithms of RWR. We describe our proposed method BePI in Section 3.3. After presenting our experimental results in Section 3.5, we summarize this work in Section 3.6.

3.2 Preliminaries

In this section, we present the preliminaries on two different approaches for RWR, iterative methods and preprocessing methods. Symbols used in the paper are summarized in Table 3.1.

Table 3.1: **Table of symbols used in Chapter 3**

Symbol	Definition
G	input graph
n	number of nodes in G
m	number of edges in G
n_1	number of spokes in G
n_2	number of hubs in G
n_3	number of deadends in G
n_{1i}	number of nodes in the ith diagonal block of $\mathbf{H_{11}}$
b	number of diagonal blocks in $\mathbf{H_{11}}$
s	seed node (=query node)
c	restart probability
k	hub selection ratio in the hub-and-spoke reordering method [98]
ε	error tolerance
$\mathbf{A}$	$(n \times n)$ adjacency matrix of G
$\mathbf{A}_{\text{nn}}$	adjacency matrix containing edges from non-deadend nodes to non-deadend nodes
$\mathbf{A}_{\text{nd}}$	adjacency matrix containing edges from non-deadend nodes to deadend nodes
$\tilde{\mathbf{A}}$	$(n \times n)$ row-normalized adjacency matrix of G
$\mathbf{H}$	$(n \times n)$ $\mathbf{H} = \mathbf{I} - (1-c)\tilde{\mathbf{A}}^\top$
$\mathbf{H}_{ij}$	$(n_i \times n_j)$ (i,j)-th partition of $\mathbf{H}$
$\mathbf{S}$	$(n_2 \times n_2)$ Schur complement of $\mathbf{H_{11}}$
$\mathbf{L}_1, \mathbf{U}_1$	$(n_1 \times n_1)$ LU factors of $\mathbf{H_{11}}$
$\tilde{\mathbf{L}}_2, \tilde{\mathbf{U}}_2$	$(n_2 \times n_2)$ incomplete LU factors of $\mathbf{S}$
$\mathbf{q}, \mathbf{q}_i$	$(n \times 1)$ starting vector, and $(n_i \times 1)$ i-th partition of $\mathbf{q}$, respectively
$\mathbf{r}, \mathbf{r}_i$	$(n \times 1)$ relevance vector, and $(n_i \times 1)$ i-th partition of $\mathbf{r}$, respectively
$\lvert\mathbf{A}\rvert$	number of non-zero entries of a matrix $\mathbf{A}$

3.2.1 Iterative Methods for RWR

Iterative methods update the RWR score vector $\mathbf{r}$ iteratively. The most well-known method is the power iteration method [13] which repeatedly updates $\mathbf{r}$ as follows:

$$\mathbf{r}^{(i)} \leftarrow (1-c)\tilde{\mathbf{A}}^{\mathbf{T}}\mathbf{r}^{(i-1)} + c\mathbf{q}$$

where $\mathbf{r}^{(i)}$ denotes the vector $\mathbf{r}$ at the i-th iteration. The repetition continues until $\mathbf{r}$ has converged (i.e., $\|\mathbf{r}^{(i)} - \mathbf{r}^{(i-1)}\|_2 \leq \varepsilon$). The vector $\mathbf{r}$ is guaranteed to converge to a unique solution if $0 < c < 1$ [83]. Krylov subspace methods [79] are also used to com-

pute the solution of a linear system shown in Equation (2.2). These methods iterate a procedure to search the solution in the Krylov subspace. Since the matrix $\mathbf{H}$ is non-singular and non-symmetric [83], any Krylov subspace method, such as GMRES [53], which handles a non-symmetric matrix, can be applied to Equation (2.2). While these iterative methods do not require preprocessing, they have expensive query cost especially when there are lots of queries, since the whole iterations need to be repeated for each query.

3.2.2 Preprocessing Methods for RWR

Many real-world applications require RWR scores of any pair of nodes, e.g., scores between two arbitrary users in social networks. Hence, quickly computing RWR queries is important and useful for real-world applications. Preprocessing methods directly calculate $\mathbf{r}$ based on precomputed results to accelerate the query speed. One naive approach is to compute $\mathbf{H}^{-1}$ as follows:

$$\mathbf{r} = c\mathbf{H}^{-1}\mathbf{q}.$$

Once $\mathbf{H}^{-1}$ is obtained in the preprocessing phase, $\mathbf{r}$ can be computed efficiently in the query phase. However, obtaining $\mathbf{H}^{-1}$ is impractical for large graphs because inverting the matrix is very time-consuming and $\mathbf{H}^{-1}$ is too dense to fit into memory. Several preprocessing methods were proposed to alleviate the problem about $\mathbf{H}^{-1}$. Fujiwara et al. [60] proposed to use matrix factorizations such as QR or LU factorization to replace $\mathbf{H}^{-1}$ (e.g., $\mathbf{H}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}$ if $\mathbf{H}$ is LU factorized). They reordered $\mathbf{H}$ based on nodes' degrees and community structures to make the inverses of factors sparse. Shin et al. [54] developed a block elimination approach called Bear which exploits a node reordering technique [98] to concentrate non-zeros of $\mathbf{H}$, and uses block

elimination [99] to compute the solution. While these preprocessing methods compute RWR queries quickly based on precomputed results, they have scalability issues for processing very large graphs because they require heavy computational cost and large memory space caused by matrix inversion inside the preprocessing phase. That is, matrix inversion requires $O(n^3)$ time and $O(n^2)$ space where n is the dimension of a matrix to be processed. Under those complexities, if n is greater than a million, it is infeasible to complete a preprocessing phase based on matrix inversion and store preprocessed data.

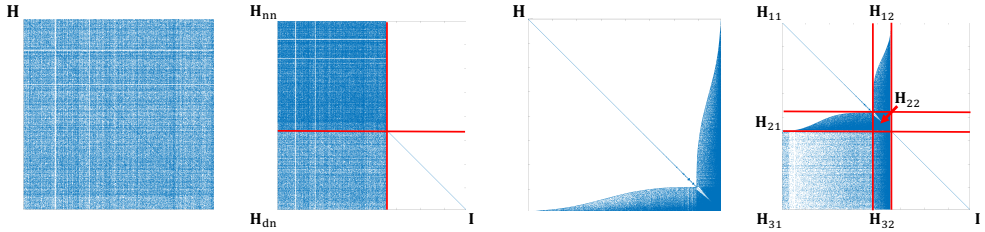
3.3 Proposed Method

In this section, we describe our proposed method BePI for fast, memory-efficient, and scalable RWR computation.

3.3.1 Overview

Preprocessing methods process relatively large graphs, and compute RWR scores quickly. However, they cannot handle very large graphs due to their high memory requirement. On the other hand, iterative methods scale to very large graphs, but show slow query speed. In this paper, our purpose is to devise a fast and scalable algorithm by taking the advantages of both preprocessing methods and iterative methods.

We present a basic version of our method BePI-B, and two optimized versions: BePI-S and BePI. BePI-B reorders nodes based on the characteristics of real-world graphs and adopts block elimination as a preprocessing method to reduce query time. Moreover, BePI-B exploits an iterative method within the block elimination approach to process very large graphs. BePI-S further improves the performance of the iterative method by sparsifying a matrix in terms of running time and memory requirement.



(a) Original matrix **H** (b) Deadend reordering (c) Hub-and-spoke reordering (d) Deadend and hub-and-spoke reordering

Figure 3.2: **Results of node reordering on the Slashdot dataset.** (a) is the original matrix **H** before node reordering. (b) and (c) are **H** reordered by deadend reordering and hub-and-spoke reordering, respectively. (d) is **H** reordered by the hub-and-spoke reordering method on top of the result of the deadend reordering method. BEPI computes RWR scores on the reordered matrix **H** in (d). $\mathbf{H}_{11}$ in (d) is a block diagonal matrix.

On top of that, BEPI-B accelerates the query speed by applying a preconditioner to the iterative method. The main ideas of our proposed method are summarized as follows:

- **BEPI-B: exploiting graph characteristics** to reorder nodes and apply block elimination (Section 3.3.2), and **incorporating an iterative method into block elimination** to increase the scalability of RWR computation (Section 3.3.3).
- **BEPI-S: sparsifying the Schur complement** to improve the performance of the iterative method (Section 3.3.4).
- **BEPI: preconditioning a linear system** to make the iterative method converge faster (Section 3.3.5).

BEPI comprises two phases: the preprocessing phase and the query phase. In the preprocessing phase (Algorithm 3), BEPI precomputes several matrices which are required by the query phase. In the query phase (Algorithm 4), BEPI computes RWR scores for each query by exploiting the precomputed matrices. Note that the preprocessing phase is run once, and the query phase is run for each seed node. To exploit sparsity of graphs, we save all matrices in a sparse matrix format such as compressed column storage [100] which stores only non-zero entries and their locations.

3.3.2 BePI-B: Exploiting Graph Characteristics for Node Reordering and Block Elimination

BEPI-B first reorders $\mathbf{H} = \mathbf{I} - (1-c)\tilde{\mathbf{A}}^{\mathbf{T}}$ based upon real-world graph characteristics, and applies block elimination for efficient RWR computation. Previous works [101, 54, 84] have shown that node reordering methods reduce computational cost of operations based on adjacency matrices of real-world graphs. For further improvement, we propose to mix node reordering strategies based on two graph characteristics: 1) deadends, and 2) hub-and-spoke structure. After reordering nodes, we apply block elimination as a preprocessing method to reduce query cost.

3.3.2.1 Node Reordering Based on Deadends and Hub-and-Spoke Structure

Deadends. Deadends are nodes having no out-going edges. Many deadends are produced from various sources such as a page containing only a file or an image in real-world graphs (see Table 3.2). Deadends have been used to improve the performance of graph operations [101]. In this paper, we reorder nodes based on deadends for efficient RWR computation. Suppose that an adjacency matrix $\mathbf{A}$ is reordered so that non-deadends and deadends are separated as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{\text{nn}} & \mathbf{A}_{\text{nd}} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

where $\mathbf{A}_{\text{nn}}$ is a submatrix containing edges from non-deadend nodes to non-deadend nodes, and $\mathbf{A}_{\text{nd}}$ is a submatrix containing edges from non-deadend nodes to deadend

nodes. Then, Equation (2.2) is represented as follows:

$$\mathbf{Hr} = c\mathbf{q} \Leftrightarrow \begin{bmatrix} \mathbf{H}_{\mathrm{nn}} & \mathbf{0} \\ \mathbf{H}_{\mathrm{dn}} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{r}_{\mathrm{n}} \\ \mathbf{r}_{\mathrm{d}} \end{bmatrix} = c \begin{bmatrix} \mathbf{q}_{\mathrm{n}} \\ \mathbf{q}_{\mathrm{d}} \end{bmatrix}$$

where $\mathbf{H}_{\mathrm{nn}} = \mathbf{I} - (1-c)\tilde{\mathbf{A}}^{\mathbf{T}}_{\mathrm{nn}}$ and $\mathbf{H}_{\mathrm{dn}} = -(1-c)\tilde{\mathbf{A}}^{\mathbf{T}}_{\mathrm{nd}}$. Figure 3.2(b) presents the example of $\mathbf{H}$ reordered by the deadend reordering approach. The partitioned solutions $\mathbf{r}_{\mathrm{n}}$ and $\mathbf{r}_{\mathrm{d}}$ are obtained from the following equations:

$$\mathbf{H}_{\mathrm{nn}}\mathbf{r}_{\mathrm{n}} = c\mathbf{q}_{\mathrm{n}} \tag{3.1}$$

$$\mathbf{r}_{\mathrm{d}} = c\mathbf{q}_{\mathrm{d}} - \mathbf{H}_{\mathrm{dn}}\mathbf{r}_{\mathrm{n}} \tag{3.2}$$

Note that the dimension and the number of non-zeros of $\mathbf{H}_{\mathrm{nn}}$ are smaller than those of $\mathbf{H}$. The partitioned solution $\mathbf{r}_{\mathrm{d}}$ is easily computed if we have $\mathbf{r}_{\mathrm{n}}$. Hence, the deadend reordering approach enables to obtain RWR scores by solving the linear system in Equation (3.1) which is smaller than the original one in Equation (2.2). One naive method for computing Equation (3.1) is to invert $\mathbf{H}_{\mathrm{nn}}$, i.e., $\mathbf{r}_{\mathrm{n}} = \mathbf{H}^{-1}_{\mathrm{nn}}\mathbf{q}_{\mathrm{n}}$. However, obtaining $\mathbf{H}^{-1}_{nn}$ is infeasible in very large graphs because its dimension is still too large to invert. To efficiently solve the linear system in Equation (3.1), we introduce another reordering technique based on the hub-and-spoke structure on top of the deadend reordering approach.

Hub-and-spoke structure. Most real-world graphs have the hub-and-spoke structure meaning they follow power-law degree distribution with few *hubs* (very high degree nodes) and majority of *spokes* (low degree nodes) [73]. The structure is exploited to concentrate entries of an adjacency matrix by reordering nodes as shown in Figure 3.2(c). The reordered matrix based on the hub-and-spoke structure

has improved the performance of operations on graphs [54]. We use the hub-and-spoke structure to efficiently solve Equation (3.1). Any reordering method based on the hub-and-spoke structure can be utilized for the purpose; in this paper, we use SlashBurn [98] because it shows the best performance in concentrating entries of an adjacency matrix (more details in Appendix A.1).

We reorder nodes of the submatrix $\mathbf{A}_{\mathrm{nn}}$ using the hub-and-spoke reordering method so that the reordered matrix contains a large but easy-to-invert submatrix such as a block diagonal one as shown in Figure 3.2(c). After reordered by the deadend approach and the hub-and-spoke reordering method, $\mathbf{H}$ is partitioned as follows:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{\mathrm{nn}} & \mathbf{0} \\ \mathbf{H}_{\mathrm{dn}} & \mathbf{I} \end{bmatrix} \Leftrightarrow \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} & \mathbf{0} \\ \mathbf{H}_{21} & \mathbf{H}_{22} & \mathbf{0} \\ \mathbf{H}_{31} & \mathbf{H}_{32} & \mathbf{I} \end{bmatrix} \tag{3.3}$$

where $\mathbf{H}_{\mathrm{nn}}$ is partitioned to $\begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{bmatrix}$, and $\mathbf{H}_{\mathrm{dn}}$ is partitioned to $\begin{bmatrix} \mathbf{H}_{31} & \mathbf{H}_{32} \end{bmatrix}$. Figure 3.2(d) illustrates the example of the reordered matrix $\mathbf{H}$ in Equation (3.3). Let n_1 be the number of spokes, n_2 be the number of hubs (see Appendix A.1), and n_3 be the number of deadends. n_1 and n_2 are determined by the hub-and-spoke reordering method, and n_3 is computed by the deadend reordering method. $\mathbf{H}_{11}$ is an $n_1 \times n_1$ matrix, and $\mathbf{H}_{22}$ is an $n_2 \times n_2$ matrix. $\mathbf{H}_{31}$ is an $n_3 \times n_1$ matrix, and $\mathbf{H}_{32}$ is an $n_3 \times n_2$ matrix. Note that $\mathbf{H}_{11}$ is block diagonal as shown in Figure 3.2(d) since $\mathbf{H}_{\mathrm{nn}}$ has the same sparsity pattern as that of the reordered matrix $\mathbf{A}_{\mathrm{nn}}^{\mathbf{T}}$ except for the diagonal entries, and the upper left part of the reordered matrix $\mathbf{A}_{\mathrm{nn}}^{\mathbf{T}}$ is a block diagonal matrix.

3.3.2.2 Block Elimination

By plugging Equation (3.3) into $\mathbf{Hr} = c\mathbf{q}$, the linear system is represented as follows:

$$\mathbf{Hr} = c\mathbf{q} \Leftrightarrow \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} & \mathbf{0} \\ \mathbf{H}_{21} & \mathbf{H}_{22} & \mathbf{0} \\ \mathbf{H}_{31} & \mathbf{H}_{32} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix} = c \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \end{bmatrix}. \tag{3.4}$$

The partitioned linear system in Equation (3.4) is solved by applying block elimination [99]. That is, the RWR solution vector $\mathbf{r}$ is obtained from the following lemma:

Lemma 3.1 (Block Elimination [99, 54])**.** *The linear system in Equation* (3.4) *is solved by block elimination, and the solution* $\mathbf{r}$ *is represented as follows:*

$$\mathbf{r} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{11}^{-1}(c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2) \\ \mathbf{S}^{-1}(c\mathbf{q}_2 - \mathbf{H}_{21}(\mathbf{H}_{11}^{-1}(c\mathbf{q}_1))) \\ c\mathbf{q}_3 - \mathbf{H}_{31}\mathbf{r}_1 - \mathbf{H}_{32}\mathbf{r}_2 \end{bmatrix} \tag{3.5}$$

where $\mathbf{S} = \mathbf{H}_{22} - \mathbf{H}_{21}\mathbf{H}_{11}^{-1}\mathbf{H}_{12}$ *is the Schur complement of* $\mathbf{H}_{11}$*. Note that the dimension of* $\mathbf{S}$ *is* $n_2 \times n_2$ *where* n_2 *is the number of hubs.*

Proof. See Section 3.4.4.2. □

If all matrices in Equation (3.5) are precomputed, the RWR score vector $\mathbf{r}$ is efficiently calculated, i.e., only matrix vector multiplications are required for the query computation.

3.3.3 BePI-B: Incorporating an Iterative Method into Block Elimination

BePI-B incorporates an iterative method within the block elimination to compute RWR on very large graphs. Based on the block elimination approach, the RWR vector $\mathbf{r} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]^\mathbf{T}$ is obtained by solving the following linear systems:

$$\mathbf{H}_{11}\mathbf{r}_1 = c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2 \tag{3.6}$$

$$\mathbf{S}\mathbf{r}_2 = c\mathbf{q}_2 - \mathbf{H}_{21}(\mathbf{H}_{11}^{-1}(c\mathbf{q}_1)) \tag{3.7}$$

$$\mathbf{r}_3 = c\mathbf{q}_3 - \mathbf{H}_{31}\mathbf{r}_1 - \mathbf{H}_{32}\mathbf{r}_2 \tag{3.8}$$

where $\mathbf{S}$ is the Schur complement of $\mathbf{H}_{11}$. Note that those equations are derived from Equation (3.5). $\mathbf{r}_3$ is easily obtained from $\mathbf{r}_1$ and $\mathbf{r}_2$ based on Equation (3.8). $\mathbf{r}_1$ is also easily computed if we have $\mathbf{r}_2$, because $\mathbf{H}_{11}$ is block diagonal and consists of small blocks; hence, $\mathbf{H}_{11}$ is easy-to-invert (i.e., $\mathbf{r}_1 = \mathbf{H}_{11}^{-1}(c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2)$). However, on very large graphs, inverting the Schur complement $\mathbf{S}$ is infeasible because the dimension of $\mathbf{S}$ is large (see Table 3.2). Hence, computing $\mathbf{r}_2$ in Equation (3.7) with $\mathbf{S}^{-1}$ is impractical on billion-scale graphs. Our solution for this problem is to exploit an iterative method to solve the linear system w.r.t. $\mathbf{r}_2$. This approach enables to avoid matrix inversion; consequently, the preprocessing time and the storage cost for $\mathbf{S}^{-1}$ are eliminated. In the preprocessing phase, BePI-B precomputes several matrices required in Equations (3.6), (3.7), and (3.8). In the query phase, BePI-B computes those equations for a given seed node based on the precomputed matrices.

BePI-B: Preprocessing phase (Algorithm 1). BePI-B first reorders the adjacency matrix $\mathbf{A}$ using the deadend reordering technique (line 1). Then, BePI-B permutes the adjacency matrix $\mathbf{A}_{\text{nn}}$ using the hub-and-spoke reordering method (details

Algorithm 1: Preprocessing phase in BePI-B and BePI-S

Input: graph: G, restart probability: c
Output: precomputed matrices: $\mathbf{L}_1^{-1}$, $\mathbf{U}_1^{-1}$, $\mathbf{S}$, $\mathbf{H}_{12}$, $\mathbf{H}_{21}$, $\mathbf{H}_{31}$ and $\mathbf{H}_{32}$.

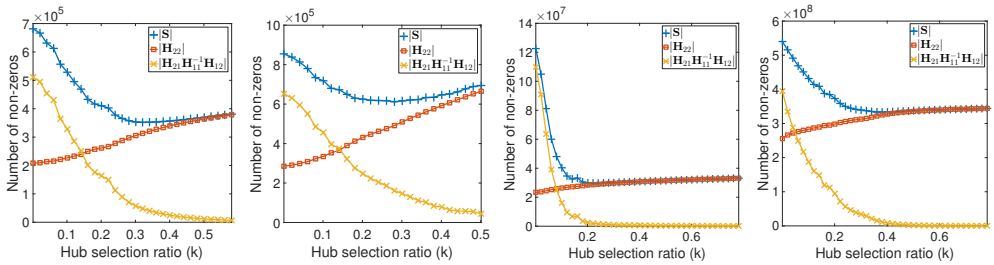
1: reorder $\mathbf{A}$ using the deadend reordering approach
2: reorder $\mathbf{A}_{nn}$ using the hub-and-spoke reordering method with the following hub selection ratio k:
   (BePI-B only) select k which makes n_2 small
   (BePI-S only) select k which minimizes $|\mathbf{S}|$
3: compute $\tilde{\mathbf{A}}$, and $\mathbf{H} = \mathbf{I} - (1-c)\tilde{\mathbf{A}}^T$
4: partition $\mathbf{H}$ into $\mathbf{H}_{11}, \mathbf{H}_{12}, \mathbf{H}_{21}, \mathbf{H}_{22}, \mathbf{H}_{31}$, and $\mathbf{H}_{32}$
5: decompose $\mathbf{H}_{11}$ into $\mathbf{L}_1$ and $\mathbf{U}_1$ using LU decomposition and compute $\mathbf{L}_1^{-1}$ and $\mathbf{U}_1^{-1}$
6: compute the Schur complement of $\mathbf{H}_{11}$, $\mathbf{S} = \mathbf{H}_{22} - \mathbf{H}_{21}(\mathbf{U}_1^{-1}(\mathbf{L}_1^{-1}(\mathbf{H}_{12})))$
7: **return** $\mathbf{L}_1^{-1}$, $\mathbf{U}_1^{-1}$, $\mathbf{S}$, $\mathbf{H}_{12}$, $\mathbf{H}_{21}$, $\mathbf{H}_{31}$, and $\mathbf{H}_{32}$

Algorithm 2: Query phase in BePI-B and BePI-S

Input: seed node: s, restart probability: c, error tolerance: ε, precomputed matrices: $\mathbf{L}_1^{-1}$, $\mathbf{U}_1^{-1}$, $\mathbf{S}$, $\mathbf{H}_{12}$, $\mathbf{H}_{21}$, $\mathbf{H}_{31}$, and $\mathbf{H}_{32}$
Output: relevance vector: $\mathbf{r}$

1: create $\mathbf{q}$ whose sth entry is 1 and the others are 0
2: partition $\mathbf{q}$ into $\mathbf{q}_1$, $\mathbf{q}_2$, and $\mathbf{q}_3$
3: compute $\tilde{\mathbf{q}}_2 = c\mathbf{q}_2 - \mathbf{H}_{21}(\mathbf{U}_1^{-1}(\mathbf{L}_1^{-1}c\mathbf{q}_1))$
4: solve $\mathbf{S}\mathbf{r}_2 = \tilde{\mathbf{q}}_2$ using an iterative method and the error tolerance ε
5: compute $\mathbf{r}_1 = \mathbf{U}_1^{-1}(\mathbf{L}_1^{-1}(c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2))$
6: compute $\mathbf{r}_3 = c\mathbf{q}_3 - \mathbf{H}_{31}\mathbf{r}_1 - \mathbf{H}_{32}\mathbf{r}_2$
7: create $\mathbf{r}$ by concatenating $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$
8: **return** $\mathbf{r}$

in Appendix A.1) so that the reordered matrix contains a large block diagonal matrix as seen in Figure 3.2(c) (line 2). Notice that when we permute $\mathbf{A}_{\text{nn}}$, the rows of $\mathbf{A}_{\text{nd}}$ also need to be permuted according to the permutation produced by the hub-and-spoke reordering method. In BePI-B, we choose a hub selection ratio k which makes the dimension of the Schur complement n_2 small enough in order to concentrate entries of $\mathbf{A}_{\text{nn}}$ as much as possible. Then, BePI-B computes and partitions $\mathbf{H}$ (lines 3 and 4). When we compute $\mathbf{H}_{11}^{-1}$, we invert the LU factors of $\mathbf{H}_{11}$ since this approach is more efficient in terms of time and space than directly inverting $\mathbf{H}_{11}$ as suggested in [60, 54] (line 5). BePI-B finally computes the Schur complement of $\mathbf{H}_{11}$ (line 6).



(a) Slashdot (b) Wikipedia (c) Flickr (d) WikiLink

Figure 3.3: **Number of non-zeros of the Schur complement** $|\mathbf{S}|$ with different hub selection ratio k on the Slashdot, the Wikipedia, the Flickr, and the WikiLink datasets. The figures show the trade-off problem for selecting k. If we select large k, then $|\mathbf{S}|$ decreases compared to small k. However, if we choose too large k (e.g., when k is greater than 0.3 in the sub-figures), then $|\mathbf{S}|$ increases. We set k between 0.2 and 0.3 since those constants decrease $|\mathbf{S}|$ enough (see Table 3.2).

BePI-B: Query phase (Algorithm 2). In the query phase, BePI-B computes the RWR score vector $\mathbf{r}$ for a given seed node s based on the precomputed matrices. The vector $\mathbf{q}$ denotes the length-n starting vector whose entry at the index of the seed node s is 1 and otherwise 0. It is partitioned into the length-n_1 vector $\mathbf{q}_1$, the length-n_2 vector $\mathbf{q}_2$, and the length-n_3 vector $\mathbf{q}_3$ (lines 1 and 2). BePI-B first solves the linear system w.r.t. $\mathbf{r}_2$ in Equation (3.7) using an iterative method (lines 3 and 4). Then, BePI-B computes $\mathbf{r}_1$ and $\mathbf{r}_3$ (lines 5 and 6).

Since $\mathbf{S}$ is non-symmetric and invertible [102], any iterative methods for a non-symmetric matrix can be used; in this paper, we use GMRES since it is the state-of-the-art method in terms of efficiency and accuracy. GMRES repeats an iteration procedure until the relative residual is less than an error tolerance ε (i.e., $|||\mathbf{S}\mathbf{r}_2^{(i)} - \tilde{\mathbf{q}}_2||_2/||\tilde{\mathbf{q}}_2||_2 \leq \varepsilon$ where $\mathbf{r}_2^{(i)}$ indicates $\mathbf{r}_2$ at the i-th iteration of GMRES).

3.3.4 BePI-S: Sparsifying the Schur Complement

We present BePI-S which improves on BePI-B by decreasing the number of non-zero entries of the Schur complement $\mathbf{S}$ used by the iterative procedure in BePI-B. Since

the time complexity of iterative methods depends on the number of non-zeros of the matrix, this approach saves time for solving the linear system on $\mathbf{S}$. Also, decreasing non-zero entries of $\mathbf{S}$ reduces the storage cost for $\mathbf{S}$. By the definition of $\mathbf{S}$ (i.e., $\mathbf{S} = \mathbf{H}_{22} - \mathbf{H}_{21}\mathbf{H}_{11}^{-1}\mathbf{H}_{12}$), the entries of $\mathbf{S}$ are determined by $\mathbf{H}_{22}$ and $\mathbf{H}_{21}\mathbf{H}_{11}^{-1}\mathbf{H}_{12}$. Thus, the number of non-zeros of $\mathbf{S}$ is roughly bounded as follows:

$$|\mathbf{S}| \leq |\mathbf{H}_{22}| + |\mathbf{H}_{21}\mathbf{H}_{11}^{-1}\mathbf{H}_{12}|$$

where $|\mathbf{A}|$ is the number of non-zeros of the matrix $\mathbf{A}$. To decrease the number of non-zeros of $\mathbf{S}$, BePI-S sets a hub selection ratio k which minimizes the number of non-zeros of $\mathbf{S}$. If we increase k, the hub-and-spoke reordering method selects more hubs at each step; therefore, n_2 increases, and n_1 decreases (i.e., $n - n_3 = n_1 + n_2$). In other words, $|\mathbf{H}_{22}|$ increases while $|\mathbf{H}_{11}|$, $|\mathbf{H}_{12}|$, and $|\mathbf{H}_{21}|$ decrease; thus, $|\mathbf{H}_{21}\mathbf{H}_{11}^{-1}\mathbf{H}_{12}|$ is also reduced. The point is that, with a suitable choice of k, $|\mathbf{S}|$ decreases since $|\mathbf{H}_{22}|$ slightly increases while $|\mathbf{H}_{21}\mathbf{H}_{11}^{-1}\mathbf{H}_{12}|$ is significantly reduced. Note that this is a trade-off problem between the number of entries of $\mathbf{H}_{22}$ and that of $\mathbf{H}_{21}\mathbf{H}_{11}^{-1}\mathbf{H}_{12}$. If we set k too large, then although $|\mathbf{H}_{21}\mathbf{H}_{11}^{-1}\mathbf{H}_{12}|$ decreases a lot, $|\mathbf{H}_{22}|$ also increases a lot; therefore, $|\mathbf{S}|$ becomes large. Figure 3.3 illustrates the trade-off problem on real-world graphs.

BePI-S: Preprocessing phase (Algorithm 1). BePI-S precomputes the matrices demanded in the query phase on top of BePI-B. First of all, BePI-S reorders $\mathbf{A}$ and $\mathbf{A}_{nn}$ using the deadend and the hub-and-spoke reordering methods similarly to BePI-B (line 1 and 2). However, when BePI-S reorders $\mathbf{A}_{nn}$ using the hub-and-spoke reordering method, we set a hub selection ratio k which minimizes the number of non-zeros of the Schur complement $\mathbf{S}$ (line 2). We empirically select k as 0.2 or 0.3

which makes the Schur complement sparse enough, as presented in Figure 3.3 and Table 3.2. As we will discuss in Section 3.5.5, BePI-S accelerates preprocessing speed by up to 10× and saves memory space by up to 5× compared to BePI-B.

BePI-S: Query phase (Algorithm 2). BePI-S computes RWR scores for a given seed node based on the precomputed matrices. Note that the query phase of BePI-S is the same as that of BePI-B. However, the query speed of BePI-S is faster than that of BePI-B because BePI-S decreases the number of non-zeros of the Schur complement used in the iterative method (line 4). As we will see in Section 3.5.5, BePI-S leads to up to 5× performance improvement in terms of query speed compared to BePI-B.

3.3.5 BePI: Preconditioning a Linear System for the Iterative Method

Our final method BePI improves BePI-S by exploiting a preconditioner [103] to enhance the speed of the iterative method in the query phase. The main purpose of preconditioning is to modify a linear system so that iterative methods converge faster. More specifically, preconditioning decreases the condition number of the matrix to be solved and makes the eigenvalues of the modified system to form a tighter cluster away from the origin. The small condition number and the tight eigenvalue distribution are the main criteria for fast convergence [53, 78]. A standard approach is to use a non-singular matrix $\mathbf{M}$ as a preconditioner. With $\mathbf{M}$, a linear system $\mathbf{Ax} = \mathbf{b}$ is preconditioned to $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$. Notice that the solution of the original system is the same as that of the preconditioned system.

BePI exploits a preconditioner to make convergence faster when solving the linear system of $\mathbf{S}$ in Equation (3.7) using an iterative method. Among various preconditioning techniques such as incomplete LU decomposition (ILU) [79] or Sparse

Approximate Inverse (SPAI) [104], we choose ILU as a preconditioner because ILU factors are easily computed and effective for preconditioning. The incomplete LU decomposition of a matrix $\mathbf{A}$ is a sparse approximation of the LU factors of the matrix, i.e., $\mathbf{A} \simeq \tilde{\mathbf{L}}\tilde{\mathbf{U}}$. The ILU factors, $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{U}}$, have the same sparsity pattern as the lower and upper triangular parts of $\mathbf{A}$, respectively.

The linear system, $\mathbf{S}\mathbf{r}_2 = \tilde{\mathbf{q}}_2$, in Equation (3.7) is preconditioned with the ILU factors of $\mathbf{S}$ as follows:

$$\tilde{\mathbf{U}}_2^{-1}\tilde{\mathbf{L}}_2^{-1}\mathbf{S}\mathbf{r}_2 = \tilde{\mathbf{U}}_2^{-1}\tilde{\mathbf{L}}_2^{-1}\tilde{\mathbf{q}}_2 \tag{3.9}$$

where $\mathbf{S} \simeq \tilde{\mathbf{L}}_2\tilde{\mathbf{U}}_2$ and $\tilde{\mathbf{q}}_2 = c\mathbf{q}_2 - \mathbf{H}_{21}(\mathbf{H}_{11}^{-1}(c\mathbf{q}_1))$. Then, an iterative method finds the solution $\mathbf{r}_2$ of the preconditioned system in Equation (3.9). However, it is difficult to explicitly construct the preconditioned system due to the inversion of the ILU factors. Instead of directly obtaining $\tilde{\mathbf{U}}_2^{-1}$ and $\tilde{\mathbf{L}}_2^{-1}$, many preconditioned iterative methods, such as preconditioned GMRES [77], involve a procedure which iteratively preconditions the original system by taking advantage of triangular matrix, $\tilde{\mathbf{L}}_2$ and $\tilde{\mathbf{U}}_2$, without explicitly constructing the preconditioned system and inverting the preconditioner (see more details in Appendix A.2). We exploit a preconditioned iterative method to solve the preconditioned system in Equation (3.9) with the preconditioner.

BePI: Preprocessing phase (Algorithm 3). BePI precomputes the matrices required for computing RWR scores in the query phase. When BePI reorders nodes using the hub-and-spoke reordering method, BePI also chooses the hub selection ratio k which minimizes the number of non-zeros of the Schur complement $\mathbf{S}$ as in BePI-S (line 2). After reordering nodes, BePI computes $\mathbf{H}$ and the Schur complement $\mathbf{S}$ (lines 3∼6). Then, BePI calculates the ILU factors of $\mathbf{S}$, $\tilde{\mathbf{L}}_2$ and $\tilde{\mathbf{U}}_2$ (line 7) to obtain a preconditioner for the iterative method in the query phase. Note that the storage cost of $\tilde{\mathbf{L}}_2$ and $\tilde{\mathbf{U}}_2$ is the same as that of $\mathbf{S}$, since $\tilde{\mathbf{L}}_2$ and $\tilde{\mathbf{U}}_2$ follow the same sparsity

Algorithm 3: Preprocessing phase in BePI

Input: graph: G, restart probability: c
Output: precomputed matrices: $\mathbf{L}_1^{-1}$, $\mathbf{U}_1^{-1}$, $\mathbf{S}$, $\tilde{\mathbf{L}}_2$, $\tilde{\mathbf{U}}_2$, $\mathbf{H}_{12}$, $\mathbf{H}_{21}$, $\mathbf{H}_{31}$ and $\mathbf{H}_{32}$.

1: reorder $\mathbf{A}$ using the deadend reordering approach
2: reorder $\mathbf{A}_{nn}$ using the hub-and-spoke reordering method with a hub selection ratio k which minimizes $|\mathbf{S}|$
3: compute $\tilde{\mathbf{A}}$, and $\mathbf{H} = \mathbf{I} - (1-c)\tilde{\mathbf{A}}^T$
4: partition $\mathbf{H}$ into $\mathbf{H}_{11}, \mathbf{H}_{12}, \mathbf{H}_{21}, \mathbf{H}_{22}, \mathbf{H}_{31}$, and $\mathbf{H}_{32}$
5: decompose $\mathbf{H}_{11}$ into $\mathbf{L}_1$ and $\mathbf{U}_1$ using LU decomposition and compute $\mathbf{L}_1^{-1}$ and $\mathbf{U}_1^{-1}$
6: compute the Schur complement of $\mathbf{H}_{11}$, $\mathbf{S} = \mathbf{H}_{22} - \mathbf{H}_{21}(\mathbf{U}_1^{-1}(\mathbf{L}_1^{-1}(\mathbf{H}_{12})))$
7: compute incomplete LU factors of $\mathbf{S} \simeq \tilde{\mathbf{L}}_2\tilde{\mathbf{U}}_2$
8: **return** $\mathbf{L}_1^{-1}$, $\mathbf{U}_1^{-1}$, $\mathbf{S}$, $\tilde{\mathbf{L}}_2$, $\tilde{\mathbf{U}}_2$, $\mathbf{H}_{12}$, $\mathbf{H}_{21}$, $\mathbf{H}_{31}$, and $\mathbf{H}_{32}$

Algorithm 4: Query phase in BePI

Input: seed node: s, restart probability: c, error tolerance: ε, precomputed matrices: $\mathbf{L}_1^{-1}$, $\mathbf{U}_1^{-1}$, $\mathbf{S}$, $\tilde{\mathbf{L}}_2$, $\tilde{\mathbf{U}}_2$, $\mathbf{H}_{12}$, $\mathbf{H}_{21}$, $\mathbf{H}_{31}$, and $\mathbf{H}_{32}$
Output: relevance vector: $\mathbf{r}$

1: create $\mathbf{q}$ whose sth entry is 1 and the others are 0
2: partition $\mathbf{q}$ into $\mathbf{q}_1$, $\mathbf{q}_2$, and $\mathbf{q}_3$
3: compute $\tilde{\mathbf{q}}_2 = c\mathbf{q}_2 - \mathbf{H}_{21}(\mathbf{U}_1^{-1}(\mathbf{L}_1^{-1}c\mathbf{q}_1))$
4: solve the preconditioned system $\tilde{\mathbf{U}}_2^{-1}\tilde{\mathbf{L}}_2^{-1}\mathbf{S}\mathbf{r}_2 = \tilde{\mathbf{U}}_2^{-1}\tilde{\mathbf{L}}_2^{-1}\tilde{\mathbf{q}}_2$ using a preconditioned iterative method with $\tilde{\mathbf{L}}_2$ and $\tilde{\mathbf{U}}_2$, and the error tolerance ε
5: compute $\mathbf{r}_1 = \mathbf{U}_1^{-1}(\mathbf{L}_1^{-1}(c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2))$
6: compute $\mathbf{r}_3 = c\mathbf{q}_3 - \mathbf{H}_{31}\mathbf{r}_1 - \mathbf{H}_{32}\mathbf{r}_2$
7: create $\mathbf{r}$ by concatenating $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$
8: **return** $\mathbf{r}$

pattern of $\mathbf{S}$.

BePI: Query phase (Algorithm 4). In the query phase, BePI computes Equations (3.6), (3.7), and (3.8) to obtain the RWR score vector $\mathbf{r}$ w.r.t. a seed node s based on the matrices precomputed by Algorithm 3. BePI first sets the starting vector $\mathbf{q}$ for given seed node s (lines 1 and 2). Then, BePI solves the preconditioned system in Equation (3.9) using an iterative method such as preconditioned GMRES (see details in Appendix A.2) with the preconditioner $\tilde{\mathbf{L}}_2$ and $\tilde{\mathbf{U}}_2$ (lines 3 and 4). After obtaining $\mathbf{r}_2$, BePI computes $\mathbf{r}_1$ and $\mathbf{r}_3$ (lines 5 and 6). As we will see in Section 3.5.5, the preconditioner accelerates query speed by up to 4× compared to BePI-S.

3.4 Theoretical Results

We analyze the time and space complexities of BePI. Moreover, we analyze the accuracy bound of BePI, since BePI exploits an iterative method. Note that all matrices are saved in a sparse matrix format such as compressed column storage [100] which contains only non-zero entries and their locations, and all sparse matrix operations such as sparse matrix vector multiplication only consider non-zero entries to exploit such sparsity.

3.4.1 Time Complexity

We provide proofs for the time complexity of BePI.

Theorem 3.1. *The preprocessing phase of BePI takes* $O(\lceil n_2/(k\times l)\rceil(m+l\log l)+\sum_{i=1}^{b} n_{1i}^3+n_2\sum_{i=1}^{b} n_{1i}^2+|\mathbf{S}|+\min(n_2^2 n_1, n_2 m))$ *where* $l = n_1+n_2$ *and* k *is the hub selection ratio of the hub-and-spoke reordering method.*

Proof. Computing $\mathbf{L}_1^{-1}$, $\mathbf{U}_1^{-1}$, and $\mathbf{S}$ after doing the hub-and-spoke reordering method takes $O(\lceil n_2/(k\times l)\rceil(m+l\log l)+\sum_{i=1}^{b} n_{1i}^3+n_2\sum_{i=1}^{b} n_{1i}^2+\min(n_2^2 n_1, n_2 m))$ [54] where $l = n_1+n_2$ and $\lceil n_2/(k\times n)\rceil$ indicates the number of iterations of SlashBurn. Since incomplete LU decomposition for a sparse matrix $\mathbf{A}$ takes $O(|\mathbf{A}|)$ [79], it takes $O(|\mathbf{S}|)$ to compute $\tilde{\mathbf{L}}_2$ and $\tilde{\mathbf{U}}_2$. □

According to Theorem 3.1, the preprocessing cost of BePI mainly depends on the number of iterations of the reordering method and the computations related to the Schur complement. Note that since the number of iterations of the hub-and-spoke reordering method [98] and $|\mathbf{S}|$ are reduced as k increases, the preprocessing cost decreases as in Figures 3.4(a) and 3.7. Also, Theorem 3.1 indicates that the preprocessing cost of BePI is much smaller than that of Bear, the state-of-the-art block elimination

approach, since BePI demands $O(|\mathbf{S}|)$ and Bear requires $O(n_2^3)$ (i.e., $|\mathbf{S}| \ll n_2^3$) while other factors are the same in both methods.

Theorem 3.2. *The query phase of BePI takes* $O(\sum_{i=1}^{b} n_{1i}^2 + \min(n_1 n_2, m) + min(n_1 n_3, m) + min(n_2 n_3, m) + T|\mathbf{S}|)$ *where* T *is the number of iterations.*

Proof. Since it takes $O(\sum_{i=1}^{b} n_{1i}^2 + \min(n_1 n_2, m))$ to compute $\tilde{\mathbf{q}}_2 = c\mathbf{q}_2 - \mathbf{H}_{21}(\mathbf{H}_{11}^{-1}(c\mathbf{q}_1))$ [54], and solving a sparse linear system $\mathbf{Ax} = \mathbf{b}$ with an iterative method takes $O(T|\mathbf{A}|)$ where T is the number of iterations [53], it takes $O(\sum_{i=1}^{b} n_{1i}^2 + \min(n_1 n_2, m) + T|\mathbf{S}|)$ to solve the linear system of $\mathbf{S}$. Note that the time complexity for computing $\mathbf{r}_1$ is the same as that of $\tilde{\mathbf{q}}_2$. For $\mathbf{r}_3$, it takes $O(min(n_1 n_3, m) + min(n_2 n_3, m))$. □

Theorem 3.2 implies that the query cost of BePI mainly depends on the number of iterations T and $|\mathbf{S}|$. Since $|\mathbf{S}|$ and T are reduced by the sparsification of the Schur complement and the preconditioner, respectively, the query cost of BePI decreases compared to those of BePI-B and BePI-S.

3.4.2 Space Complexity

We provide a proof for the space complexity of BePI.

Theorem 3.3. *BePI requires* $O(\sum_{i=1}^{b} n_{1i}^2 + \min(n_1 n_2, m) + min(n_1 n_3, m) + min(n_2 n_3, m) + |\mathbf{S}|)$ *memory space for preprocessed matrices:* $\mathbf{L}_1^{-1}$, $\mathbf{U}_1^{-1}$, $\mathbf{S}$, $\tilde{\mathbf{L}}_2$, $\tilde{\mathbf{U}}_2$, $\mathbf{H}_{12}$, $\mathbf{H}_{21}$, $\mathbf{H}_{31}$, *and* $\mathbf{H}_{32}$.

Proof. It requires $O(min(n_1 n_2, m))$ memory space for $\mathbf{H}_{12}$ and $\mathbf{H}_{12}$, and $O(\sum_{i=1}^{b} n_{1i}^2)$ memory space for $\mathbf{L}_1^{-1}$ and $\mathbf{U}_1^{-1}$ [54]. Also, it requires $O(min(n_1 n_3, m) + min(n_2 n_3, m))$ memory space for $\mathbf{H}_{31}$ and $\mathbf{H}_{32}$. Since the space cost for incomplete LU factors is the same as that of the given sparse matrix, it requires $O(|\mathbf{S}|)$ for $\mathbf{S}$, $\tilde{\mathbf{L}}_2$ and $\tilde{\mathbf{U}}_2$. □

Theorem 3.3 indicates that the space cost of BePI mainly depends on $O(|\mathbf{S}|)$ because the number of non-zeros of $\mathbf{S}$ is larger than those of other matrices except for the incomplete LU factors of $\mathbf{S}$. Note that BePI demands much smaller memory space than the state-of-the-art method Bear because the space cost of Bear mainly depends on $O(n_2^2)$; i.e., $|\mathbf{S}| \ll n_2^2$. Also, through sparsifying the Schur complement $\mathbf{S}$, the space costs of BePI and BePI-S decrease compared to that of BePI-B which is the basic version without sparsifying $\mathbf{S}$.

3.4.3 Accuracy Bound

We analyze the accuracy bound of the RWR score vector $\mathbf{r}$ computed by BePI. Since $\mathbf{r}$ consists of $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$, and $\mathbf{r}_1$ and $\mathbf{r}_3$ are computed after $\mathbf{r}_2$, we first analyze the bound of $\mathbf{r}_2$ in Lemma 3.2, that of $\mathbf{r}_1$ in Lemma 3.3, and that of $\mathbf{r}_3$ in Lemma 3.4. Then, we conclude the bound of $\mathbf{r}$ in Theorem 3.4 using these lemmas.

Lemma 3.2 (Accuracy Bound of $\mathbf{r}_2$)**.** *Let $\mathbf{r}_2^*$ be the true solution of the linear system $\mathbf{S}\mathbf{r}_2 = \tilde{\mathbf{q}}_2$ where $\tilde{\mathbf{q}}_2 = c\mathbf{q}_2 - \mathbf{H}_{21}(\mathbf{H}_{11}^{-1}(c\mathbf{q}_1))$, and $\mathbf{r}_2^{(k)}$ be the solution computed by BePI after the relative residual becomes less than a given tolerance ε at the k-th iteration. Then, $||\mathbf{r}_2^* - \mathbf{r}_2^{(k)}||_2 \leq \frac{||\tilde{\mathbf{q}}_2||_2}{\sigma_{min}(\mathbf{S})}\varepsilon$ where $\sigma_{min}(\mathbf{S})$ is the smallest singular value of $\mathbf{S}$.*

Proof. See Section 3.4.4.3. □

Lemma 3.3 (Accuracy Bound of $\mathbf{r}_1$)**.** *Let $\mathbf{r}_1^*$ be the true solution of the linear system $\mathbf{H}_{11}\mathbf{r}_1 = \tilde{\mathbf{q}}_1$ where $\tilde{\mathbf{q}}_1 = c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2$, and $\mathbf{r}_1^{(k)}$ be the solution of $\mathbf{H}_{11}\mathbf{r}_1^{(k)} = \tilde{\mathbf{q}}_1^{(k)}$ where $\tilde{\mathbf{q}}_1^{(k)} = c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2^{(k)}$. Then, $||\mathbf{r}_1^* - \mathbf{r}_1^{(k)}||_2 \leq \frac{||\mathbf{H}_{12}||_2}{\sigma_{min}(\mathbf{H}_{11})}||\mathbf{r}_2^* - \mathbf{r}_2^{(k)}||_2 \leq \frac{||\mathbf{H}_{12}||_2||\tilde{\mathbf{q}}_2||_2}{\sigma_{min}(\mathbf{H}_{11})\sigma_{min}(\mathbf{S})}\varepsilon$ where $\sigma_{min}(\mathbf{A})$ is the smallest singular value of a matrix $\mathbf{A}$.*

Proof. See Section 3.4.4.4. □

Lemma 3.4 (Accuracy Bound of $\mathbf{r}_3$). *Let $\mathbf{r}_3^*$ be the true solution of the equation $\mathbf{r}_3 = c\mathbf{q}_3 - \mathbf{H}_{31}\mathbf{r}_1^* - \mathbf{H}_{32}\mathbf{r}_2^*$, and $\mathbf{r}_3^{(k)}$ be the solution of $\mathbf{r}_3^{(k)} = c\mathbf{q}_3 - \mathbf{H}_{31}\mathbf{r}_1^{(k)} - \mathbf{H}_{32}\mathbf{r}_2^{(k)}$. Then, $\|\mathbf{r}_3^* - \mathbf{r}_3^{(k)}\|_2 \leq \|\mathbf{H}_{31}\|_2\|\mathbf{r}_1^* - \mathbf{r}_1^{(k)}\|_2 + \|\mathbf{H}_{32}\|_2\|\mathbf{r}_2^* - \mathbf{r}_2^{(k)}\|_2$.*

Proof. See Section 3.4.4.5. □

Theorem 3.4 (Accuracy Bound of BEPI). *Let $\mathbf{r}^*$ be the true solution of the linear system $\mathbf{Hr} = c\mathbf{q}$, and $\mathbf{r}^{(k)}$ be the solution $\mathbf{r}^{(k)} = \left[\mathbf{r}_1^{(k)}, \mathbf{r}_2^{(k)}, \mathbf{r}_3^{(k)}\right]^{\mathbf{T}}$ where $\mathbf{r}_2^{(k)}$ is the solution of $\mathbf{Sr}_2 = \tilde{\mathbf{q}}_2$ computed by BEPI after the residual becomes less than the error tolerance ε at the k-th iteration, $\mathbf{r}_1^{(k)}$ is the solution of $\mathbf{H}_{11}\mathbf{r}_1^{(k)} = c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2^{(k)}$, and $\mathbf{r}_3^{(k)}$ is the solution of $\mathbf{r}_3 = c\mathbf{q}_3 - \mathbf{H}_{31}\mathbf{r}_1^{(k)} - \mathbf{H}_{32}\mathbf{r}_2^{(k)}$. Let $\alpha = \frac{||\mathbf{H}_{12}||_2}{\sigma_{min}(\mathbf{H}_{11})}$. Then, $||\mathbf{r}^* - \mathbf{r}^{(k)}||_2$ is bounded as follows:*

$$||\mathbf{r}^* - \mathbf{r}^{(k)}||_2 \leq \left(\sqrt{(\alpha||\mathbf{H}_{31}||_2 + ||\mathbf{H}_{32}||_2)^2 + \alpha^2 + 1}\right) \frac{||\tilde{\mathbf{q}}_2||_2}{\sigma_{min}(\mathbf{S})}\varepsilon.$$

Proof. By the definition of L2-norm, $||\mathbf{r}^* - \mathbf{r}^{(k)}||_2^2$ is represented as follows:

$$||\mathbf{r}^* - \mathbf{r}^{(k)}||_2^2 = \left\| \begin{bmatrix} \mathbf{r}_1^* - \mathbf{r}_1^{(k)} \\ \mathbf{r}_2^* - \mathbf{r}_2^{(k)} \\ \mathbf{r}_3^* - \mathbf{r}_3^{(k)} \end{bmatrix} \right\|_2^2 = \|\mathbf{r}_1^* - \mathbf{r}_1^{(k)}\|_2^2 + \|\mathbf{r}_2^* - \mathbf{r}_2^{(k)}\|_2^2 + \|\mathbf{r}_3^* - \mathbf{r}_3^{(k)}\|_2^2$$

Then, by Lemma 3.4, it is bounded as follows:

$$\begin{aligned} ||\mathbf{r}^* - \mathbf{r}^{(k)}||_2^2 \leq & \|\mathbf{r}_1^* - \mathbf{r}_1^{(k)}\|_2^2 + \|\mathbf{r}_2^* - \mathbf{r}_2^{(k)}\|_2^2 + \|\mathbf{H}_{31}\|_2^2\|\mathbf{r}_1^* - \mathbf{r}_1^{(k)}\|_2^2 \\ & + \|\mathbf{H}_{32}\|_2^2\|\mathbf{r}_2^* - \mathbf{r}_2^{(k)}\|_2^2 + 2\|\mathbf{H}_{31}\|_2\|\mathbf{H}_{32}\|_2\|\mathbf{r}_1^* - \mathbf{r}_1^{(k)}\|_2\|\mathbf{r}_2^* - \mathbf{r}_2^{(k)}\|_2 \end{aligned}$$

From Lemma 3.3, $||\mathbf{r}_1^* - \mathbf{r}_1^{(k)}||_2 \leq \frac{||\mathbf{H}_{12}||_2}{\sigma_{min}(\mathbf{H}_{11})}||\mathbf{r}_2^* - \mathbf{r}_2^{(k)}||_2 = \alpha||\mathbf{r}_2^* - \mathbf{r}_2^{(k)}||_2$ where $\alpha =$

$\frac{||\mathbf{H}_{12}||_2}{\sigma_{min}(\mathbf{H}_{11})}$. Hence, the bound is represented as follows:

$$
\begin{aligned}
||\mathbf{r}^* - \mathbf{r}^{(k)}||_2^2 &\leq \|\mathbf{r}_2^* - \mathbf{r}_2^{(k)}\|_2^2 \Big(\alpha^2 + 1 + \alpha^2\|\mathbf{H}_{31}\|_2^2 + \|\mathbf{H}_{32}\|_2^2 + 2\alpha\|\mathbf{H}_{31}\|_2\|\mathbf{H}_{32}\|_2\Big) \\
&= ||\mathbf{r}_2^* - \mathbf{r}_2^{(k)}||_2^2 \left(\alpha^2 + 1 + (\alpha\|\mathbf{H}_{31}\|_2 + \|\mathbf{H}_{32}\|_2)^2\right).
\end{aligned}
$$

By Lemma 3.2, $\|\mathbf{r}_2^* - \mathbf{r}_2^{(k)}\|_2 \leq \frac{||\tilde{\mathbf{q}}_2||_2}{\sigma_{min}(\mathbf{S})}\varepsilon$; thus, the above inequality is written as:

$$
||\mathbf{r}^* - \mathbf{r}^{(k)}||_2^2 \leq \left(\alpha^2 + 1 + (\alpha\|\mathbf{H}_{31}\|_2 + \|\mathbf{H}_{32}\|_2)^2\right)\left(\frac{\|\tilde{\mathbf{q}}_2\|_2}{\sigma_{min}(\mathbf{S})}\varepsilon\right)^2.
$$

Finally, the bound of $||\mathbf{r}^* - \mathbf{r}^{(k)}||_2$ is represented in the following inequality:

$$
||\mathbf{r}^* - \mathbf{r}^{(k)}||_2 \leq \left(\sqrt{(\alpha||\mathbf{H}_{31}||_2 + ||\mathbf{H}_{32}||_2)^2 + \alpha^2 + 1}\right)\frac{||\tilde{\mathbf{q}}_2||_2}{\sigma_{min}(\mathbf{S})}\varepsilon
$$

□

According to Theorem 3.4, the accuracy of BePI is bounded by the norms and the smallest singular values of the input matrices and the error tolerance ε. Also, Theorem 3.4 indicates that BePI guarantees $||\mathbf{r}^* - \mathbf{r}^{(k)}||_2 \leq \varepsilon_T$ where ε_T is the target accuracy if we set the error tolerance to ε satisfying the following inequality:

$$
0 < \varepsilon \leq \left(\sqrt{(\alpha||\mathbf{H}_{31}||_2 + ||\mathbf{H}_{32}||_2)^2 + \alpha^2 + 1}\right)^{-1}\frac{\sigma_{min}(\mathbf{S})}{||\tilde{\mathbf{q}}_2||_2}\varepsilon_T.
$$

3.4.4 Lemmas and Proofs

We describe the lemmas and proofs used in the above theoretical results.

3.4.4.1 Proof of Inverse Inequality

Lemma 3.5. *For* $\mathbf{Ax} = \mathbf{b}$*, if a matrix* $\mathbf{A}$ *is invertible, then* $||\mathbf{A}^{-1}||_2^{-1}||\mathbf{x}||_2 \leq ||\mathbf{Ax}||_2$.

Proof. Since $\mathbf{A}$ is invertible, $\|\mathbf{x}\|_2$ is bounded as follows:

$$\|\mathbf{x}\|_2 = \|\mathbf{A}^{-1}\mathbf{A}\mathbf{x}\|_2 \leq \|\mathbf{A}^{-1}\|_2\|\mathbf{A}x\|_2.$$

Hence, $||\mathbf{A}^{-1}||_2^{-1}||\mathbf{x}||_2 \leq ||\mathbf{A}\mathbf{x}||_2$. □

3.4.4.2 Proof of Lemma 3.1

Proof. The partitioned linear system in Equation (3.4) is represented using Equations (3.1) and (3.2) as follows:

$$\begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{21} & \mathbf{H}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix} = c \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \end{bmatrix} \tag{3.10}$$

$$\mathbf{r}_3 = c\mathbf{q}_3 - \mathbf{H}_{31}\mathbf{r}_1 - \mathbf{H}_{32}\mathbf{r}_2$$

Equation (3.10) is split into two equations:

$$\mathbf{H}_{11}\mathbf{r}_1 + \mathbf{H}_{12}\mathbf{r}_2 = c\mathbf{q}_1 \tag{3.11}$$

$$\mathbf{H}_{21}\mathbf{r}_1 + \mathbf{H}_{22}\mathbf{r}_2 = c\mathbf{q}_2 \tag{3.12}$$

Then, $\mathbf{r}_1$ is obtained from Equation (3.11) as follows:

$$\begin{aligned} \mathbf{H}_{11}\mathbf{r}_1 + \mathbf{H}_{12}\mathbf{r}_2 = c\mathbf{q}_1 &\Rightarrow \mathbf{H}_{11}\mathbf{r}_1 = c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2 \\ &\Rightarrow \mathbf{r}_1 = \mathbf{H}_{11}^{-1}(c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2). \end{aligned}$$

If we plug the above equation of $\mathbf{r}_1$ into Equation (3.12), then it is represented as follows:

$$\begin{aligned}
&\mathbf{H}_{21}\mathbf{r}_1 + \mathbf{H}_{22}\mathbf{r}_2 = c\mathbf{q}_2 \\
&\Rightarrow \mathbf{H}_{21}(\mathbf{H}_{11}^{-1}(c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2)) + \mathbf{H}_{22}\mathbf{r}_2 = c\mathbf{q}_2 \\
&\Rightarrow \mathbf{H}_{21}(\mathbf{H}_{11}^{-1}(c\mathbf{q}_1)) + (\mathbf{H}_{22} - \mathbf{H}_{21}\mathbf{H}_{11}^{-1}\mathbf{H}_{12})\mathbf{r}_2 = c\mathbf{q}_2 \\
&\Rightarrow \mathbf{S}\mathbf{r}_2 = c\mathbf{q}_2 - \mathbf{H}_{21}(\mathbf{H}_{11}^{-1}(c\mathbf{q}_1)) \\
&\Rightarrow \mathbf{r}_2 = \mathbf{S}^{-1}(c\mathbf{q}_2 - \mathbf{H}_{21}(\mathbf{H}_{11}^{-1}(c\mathbf{q}_1)))
\end{aligned}$$

where $\mathbf{S} = \mathbf{H}_{22} - \mathbf{H}_{21}\mathbf{H}_{11}^{-1}\mathbf{H}_{12}$. Note that $\mathbf{H}$ and $\mathbf{H}_{11}$ is invertible if $0 < c < 1$ since they are strictly diagonally dominant; $\mathbf{S}$ is invertible because $\mathbf{H}$ is invertible [102]. □

3.4.4.3 Proof of Lemma 3.2

Proof. Since we use GMRES to solve the linear system of $\mathbf{S}$, we first analyze the accuracy bound of GMRES. Since GMRES stops the iteration when the relative residual $\frac{||\mathbf{S}\mathbf{r}_2^{(k)} - \tilde{\mathbf{q}}_2||_2}{||\tilde{\mathbf{q}}_2||_2} \leq \varepsilon$, the inequality is written as follows:

$$||\mathbf{S}\mathbf{r}_2^{(k)} - \tilde{\mathbf{q}}_2||_2 \leq \varepsilon||\tilde{\mathbf{q}}_2||_2 \Rightarrow ||\mathbf{S}\mathbf{r}_2^{(k)} - \mathbf{S}\mathbf{r}_2^*||_2 \leq \varepsilon||\tilde{\mathbf{q}}_2||_2 \Rightarrow ||\mathbf{S}(\mathbf{r}_2^* - \mathbf{r}_2^{(k)})||_2 \leq \varepsilon||\tilde{\mathbf{q}}_2||_2$$

Note that $\mathbf{H}$ and $\mathbf{H}_{11}$ are invertible because those matrices are diagonally dominant; this fact implies that $\mathbf{S}$ is invertible [102], and we are able to apply Lemma 3.5 to the

last equation as follows:

$$
\begin{aligned}
&||\mathbf{S}^{-1}||_2^{-1}||\mathbf{r}_2^* - \mathbf{r}_2^{(k)}||_2 \leq ||\mathbf{S}(\mathbf{r}_2^* - \mathbf{r}_2^{(k)})||_2 \leq \varepsilon||\tilde{\mathbf{q}}_2||_2 \\
\Rightarrow\ &||\mathbf{S}^{-1}||_2^{-1}||\mathbf{r}_2^* - \mathbf{r}_2^{(k)}||_2 \leq \varepsilon||\tilde{\mathbf{q}}_2||_2 \\
\Rightarrow\ &||\mathbf{r}_2^* - \mathbf{r}_2^{(k)}||_2 \leq \varepsilon||\mathbf{S}^{-1}||_2||\tilde{\mathbf{q}}_2||_2
\end{aligned}
$$

Since $||\mathbf{S}^{-1}||_2 = \sigma_{min}(\mathbf{S})^{-1}$ [105], $||\mathbf{r}_2^{(k)} - \mathbf{r}_2^*||_2$ is bounded as follows:

$$
||\mathbf{r}_2^* - \mathbf{r}_2^{(k)}||_2 \leq \frac{||\tilde{\mathbf{q}}_2||_2}{\sigma_{min}(\mathbf{S})}\varepsilon
$$

where $\sigma_{min}(\mathbf{S})$ is the smallest singular value of $\mathbf{S}$. □

3.4.4.4 Proof of Lemma 3.3

Proof. Since $\mathbf{H}_{11}\mathbf{r}_1^* = c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2^*$ and $\mathbf{H}_{11}\mathbf{r}_1^{(k)} = c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2^{(k)}$, $||\mathbf{H}_{11}\mathbf{r}_1^* - \mathbf{H}_{11}\mathbf{r}_1^{(k)}||_2$ is represented as follows:

$$
\begin{aligned}
||\mathbf{H}_{11}\mathbf{r}_1^* - \mathbf{H}_{11}\mathbf{r}_1^{(k)}||_2 &= ||c\mathbf{q}_1 - \mathbf{H}_{12}\mathbf{r}_2^* - c\mathbf{q}_1 + \mathbf{H}_{12}\mathbf{r}_2^{(k)}||_2 \\
&= ||\mathbf{H}_{12}\mathbf{r}_2^* - \mathbf{H}_{12}\mathbf{r}_2^{(k)}||_2 \\
&= ||\mathbf{H}_{12}(\mathbf{r}_2^* - \mathbf{r}_2^{(k)})||_2
\end{aligned}
$$

Since L2-norm is a sub-multiplicative norm [105], $||\mathbf{H}_{12}(\mathbf{r}_2^* - \mathbf{r}_2^{(k)})||_2 \leq ||\mathbf{H}_{12}||_2||\mathbf{r}_2^* - \mathbf{r}_2^{(k)}||_2$. Hence $||\mathbf{H}_{11}\mathbf{r}_1^* - \mathbf{H}_{11}\mathbf{r}_1^{(k)}||_2$ is bounded as follows:

$$
||\mathbf{H}_{11}\mathbf{r}_1^* - \mathbf{H}_{11}\mathbf{r}_1^{(k)}||_2 \leq ||\mathbf{H}_{12}||_2||\mathbf{r}_2^* - \mathbf{r}_2^{(k)}||_2
$$

By Lemma 3.5, $(||\mathbf{H}_{11}^{-1}||_2)^{-1}||\mathbf{r}_1^* - \mathbf{r}_1^{(k)}||_2 \leq ||\mathbf{H}_{11}(\mathbf{r}_1^* - \mathbf{r}_1^{(k)})||_2$. Hence, $||\mathbf{r}_1^* - \mathbf{r}_1^{(k)}||_2$ is bounded as follows:

$$||\mathbf{r}_1^* - \mathbf{r}_1^{(k)}||_2 \leq ||\mathbf{H}_{11}^{-1}||_2||\mathbf{H}_{12}||_2||\mathbf{r}_2^* - \mathbf{r}_2^{(k)}||_2.$$

By Lemma 3.2, $||\mathbf{r}_2^* - \mathbf{r}_2^{(k)}||_2 \leq \frac{||\tilde{\mathbf{q}}_2||_2}{\sigma_{min}(\mathbf{S})}\varepsilon$, and $||\mathbf{H}_{11}^{-1}||_2 = \sigma_{min}(\mathbf{H}_{11})^{-1}$ [105]. Therefore, $||\mathbf{r}_1^* - \mathbf{r}_1^{(k)}||_2$ is bounded as follows:

$$||\mathbf{r}_1^* - \mathbf{r}_1^{(k)}||_2 \leq \frac{||\mathbf{H}_{12}||_2||\tilde{\mathbf{q}}_2||_2}{\sigma_{min}(\mathbf{H}_{11})\sigma_{min}(\mathbf{S})}\varepsilon.$$

□

3.4.4.5 Proof of Lemma 3.4

Proof. From the triangular inequality and the submultiplicative property of L2-norm, it is represented as follows:

$$\begin{aligned}
\|\mathbf{r}_3^* - \mathbf{r}_3^{(k)}\|_2 &= \|-\mathbf{H}_{31}\mathbf{r}_1^* - \mathbf{H}_{32}\mathbf{r}_2^* + \mathbf{H}_{31}\mathbf{r}_1^{(k)} + \mathbf{H}_{32}\mathbf{r}_2^{(k)}\|_2 \\
&= \|\mathbf{H}_{31}(\mathbf{r}_1^* - \mathbf{r}_1^{(k)}) + \mathbf{H}_{32}(\mathbf{r}_2^* - \mathbf{r}_2^{(k)})\|_2 \\
&\leq \|\mathbf{H}_{31}(\mathbf{r}_1^* - \mathbf{r}_1^{(k)})\|_2 + \|\mathbf{H}_{32}(\mathbf{r}_2^* - \mathbf{r}_2^{(k)})\|_2 \\
&\leq \|\mathbf{H}_{31}\|_2\|\mathbf{r}_1^* - \mathbf{r}_1^{(k)}\|_2 + \|\mathbf{H}_{32}\|_2\|\mathbf{r}_2^* - \mathbf{r}_2^{(k)}\|_2
\end{aligned}$$

□

3.5 Experiments

In this section, we evaluate the performance of our method BePI, and compare it to other existing methods for computing RWR scores. We aim to answer the following questions from the experiments:

- **Q1. Preprocessing cost (Section 3.5.2).** How much memory space do BePI and other methods require for their preprocessed results? How long does this preprocessing phase take?
- **Q2. Query cost (Section 3.5.3).** How quickly does BePI respond to an RWR query compared to other methods?
- **Q3. Scalability (Section 3.5.4).** How well does BePI scale up compared to other methods?
- **Q4. Effectiveness of the optimizations (Section 3.5.5).** How effective are the sparsification of the Schur complement (Section 3.3.4) and the preconditioning (Section 3.3.5) in terms of preprocessing and query cost?
- **Q5. Effects of the hub selection ratio k (Section 3.5.6).** How does the hub selection ratio k in Algorithm 3 affect the performance of BePI in terms of running time and memory requirement?
- **Q6. Accuracy (Section 3.5.7).** Does BePI produce the exact result for the RWR computation compared to other methods?
- **Q7. Detailed comparison to Bear (Section 3.5.8).** How much does BePI improve the computational performance for RWR compared to Bear, the state-of-the-art method?

3.5.1 Experimental Settings

Machine. All experiments are conducted on a workstation with a single CPU Intel(R) Xeon(R) CPU E7540 @ 2.00GHz and 500GB memory.

Methods. We compare our methods with power iteration, LU decomposition, a Krylov subspace method (GMRES), and Bear, all of which are described in Section 3.2. We evaluate our approach using three different versions:

- BePI-B is the basic version without the sparsification of the Schur complement and the preconditioner.
- BePI-S exploits only the sparsification of the Schur complement without the preconditioner.
- BePI uses both the sparsification of the Schur complement and the preconditioner.

Approximate methods are excluded from the experiments since all the aforementioned methods including our methods compute exact RWR scores. All these methods are implemented in C++ and Eigen[1] which is an open source C++ numerical linear algebra package.

Parameters. We set the restart probability c to 0.05 as in the previous works [82, 54]. For Bear and BePI-B, we set k of the hub-and-spoke reordering method to 0.001 as in the previous work [54]. For BePI-S and BePI, we set k of the hub-and-spoke reordering method differently for each dataset as described in Table 3.2 to make the Schur complement sparse. For larger graphs, 0.2 is usually used for k. The error tolerance ε for power iteration, GMRES, and our method is set to 10^{-9}. We set the time limit for preprocessing to 24 hours.

[1] **http://eigen.tuxfamily.org/**

Table 3.2: **Summary of real-world datasets** where n is the number of nodes, m is the number of edges, and k is the hub selection ratio in SlashBurn used for BePI-S and BePI. n_1 is the number of spokes, n_2 is the number of hubs, and n_3 is the number of deadends. For BePI-B, we set k to 0.001 in SlashBurn. Note that n_2 of BePI-S are the same as that of BePI.

dataset	n	m	k	n_1 **in BePI-B**	n_1 **in BePI and BePI-S**	n_2 **in BePI-B**	n_2 **in BePI and BePI-S**	n_3
Slashdot	79,120	515,581	0.30	37,872	31,920	7,728	13,680	33,520
Wikipedia	100,312	1,627,472	0.25	79,737	72,187	16,512	24,062	4,063
Baidu	415,641	3,284,317	0.20	347,596	315,586	46,886	78,896	21,159
Flickr	2,302,925	33,140,017	0.20	1,717,120	1,554,006	225,388	388,502	360,417
LiveJournal	4,847,571	68,475,391	0.30	3,138,041	2,655,345	1,156,291	1,638,987	553,239
WikiLink	11,196,007	340,240,450	0.20	8,670,438	8,062,003	2,505,984	3,114,419	19,585
Twitter	41,652,230	1,468,365,182	0.20	33,927,419	24,061,969	6,175,862	16,041,312	1,548,949
Friendster	68,349,466	2,586,147,869	0.20	43,666,118	33,666,118	12,444,080	22,444,080	12,239,268

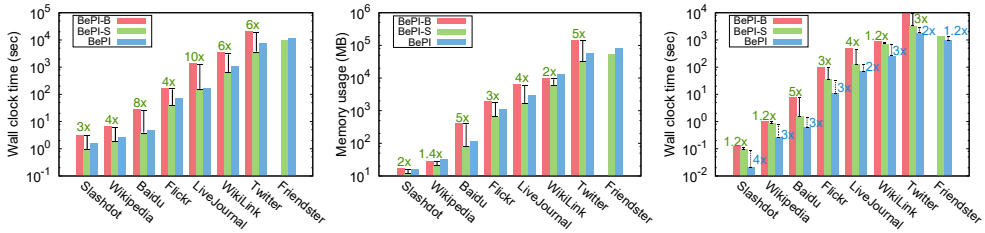
Data. The graph data used in our experiments are summarized in Table 3.2. Each dataset is briefly described as follows:

- **Slashdot**[2]**.** This is the social network in the technology news site Slashdot.
- **Wikipedia**[3]**.** This is the small network between articles of the English Wikipedia.
- **Baidu**[4]**.** This is the hyperlink network between articles of the Chinese online encyclopedia Baidu.
- **Flickr**[5]**.** This is the friendship network of Flickr users.
- **LiveJournal**[6]**.** Nodes are users of LiveJournal, and directed edges represent friendships.
- **WikiLink**[7]**.** This network consists of the wiki-links of the English Wikipedia.
- **Twitter**[8]**.** This is the follower network from Twitter, containing 1.4 billion directed follow edges between 41 million Twitter users.
- **Friendster**[9]**.** This is the friendship network of the online social site Friendster.

3.5.2 Preprocessing Cost

We examine the cost of the preprocessing phase of BePI in terms of preprocessing time and memory space for preprocessed data. We compare our method with Bear and LU decomposition, the best preprocessing methods. Preprocessing time is measured in wall-clock time, and it includes the time taken for SlashBurn in BePI and Bear.

[2] **http://dai-labor.de/IRML/datasets**
[3] **http://konect.uni-koblenz.de/networks/link-dynamic-simplewiki**
[4] **http://zhishi.me**
[5] **http://socialnetworks.mpi-sws.org/data-wosn2008.html**
[6] **http://snap.stanford.edu/data/soc-LiveJournal1.html**
[7] **http://dumps.wikimedia.org/**
[8] **http://an.kaist.ac.kr/traces/WWW2010.html**
[9] **https://archive.org/details/friendster-dataset-201107**



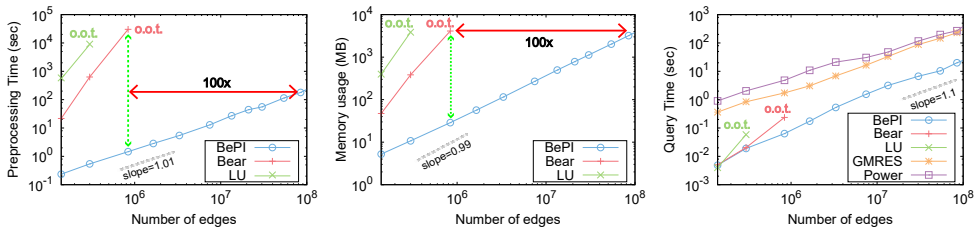
(a) Effect on preprocessing time

(b) Effect on space for preprocessed data

(c) Effect on query time

Figure 3.4: **Effect of the sparsification of the Schur complement and the preconditioning**. In these figures, bars are omitted in case the corresponding experiments run out of memory. In terms of the effect of the sparsification of the Schur complement, (a) and (b) show that the preprocessing cost is reduced: BePI-S is up to 10× faster than BePI-B, and BePI-S requires up to 5× less memory space than BePI-B. Moreover, (c) presents that the query time is also decreased: BePI-S is up to 5× faster than BePI-B in the query phase. In terms of the effect of the preconditioning, the preprocessing cost of BePI is slightly larger than that of BePI-S as seen in (a) and (b) due to the additional operation for incomplete LU factors. However, BePI is up to 4× faster than BePI-S in the query phase thanks to the effect of the preconditioning as shown in (c).

Figures 3.1(a) and 3.1(b) show the preprocessing time and the memory space usage of preprocessed data. Note that only BePI successfully performs the preprocessing phase for all the datasets, while other methods fail because their memory requirements are high, or they run out of time. As seen in Figure 3.1(a), BePI requires the least amount of time, which is less than about 2 hours for all the datasets. For the Slashdot dataset, which is the smallest dataset, BePI is $3,679\times$ faster than Bear. For other datasets, Bear and LU decomposition fail to show the results (they took more than 24 hours). To compare memory efficiency, we measure how much memory each method requires for the preprocessed matrices. As seen in Figure 3.1(b), BePI requires the least amount of space for preprocessed matrices. BePI requires up to $130\times$ less memory space than other competitors in all the datasets, which indicates the superiority of our method in terms of scalability compared to other preprocessing methods.



(a) Preprocessing time (b) Space for preprocessed data (c) Query time

Figure 3.5: **Scalability of BePI** compared to other methods on the WikiLink dataset. (a), (b), and (c) show the scalability of the three methods in terms of the number of edges. o.o.t. stands for out of time (more than 24 hours). BePI shows up to 100× better scalability than existing preprocessing methods, and scales well with regard to the size of graphs. Also, BePI provides near linear scalability in terms of preprocessing and query cost.

3.5.3 Query Cost

We compare BePI with other methods in terms of query cost. We compare our method with power iteration, GMRES, Bear, and LU decomposition. We measure the average query time for 30 random seed nodes.

As presented in Figure 3.1(c), only BePI and iterative methods successfully compute RWR scores on all the datasets, and BePI outperforms competitors for large graphs. For the Baidu dataset, BePI is up to 9× faster than GMRES, which is the second best one. For the largest Friendster dataset, BePI is 3× faster than GMRES. Compared to power iteration, BePI is 19× and 10× faster for the Baidu and the Friendster datasets, respectively.

3.5.4 Scalability

We compare the scalability of BePI against existing methods, in terms of the number of edges. For the WikiLink dataset, we extract the principal submatrices, which are the upper left part of the adjacency matrix, of different lengths so that the number of edges of each matrix is different. For each submatrix, we preprocess the matrix using

BePI, Bear, and LU decomposition. Then, we compute RWR scores using BePI, Bear, LU decomposition, power iteration, and GMRES. We measure preprocessing time, memory usage and average query time for 30 randomly selected seed nodes.

Figure 3.5 presents that BePI shows a good scalability with respect to the number of edges, while other preprocessing methods fail to scale up. As shown in Figures 3.5(a) and 3.5(b), BePI processes 100× larger graph, while using less memory space than other preprocessing methods. Also, the slope of the fitted line for BePI is 1.01 in Figure 3.5(a), 0.99 in Figure 3.5(b), and 1.1 in Figure 3.5(c). These results indicate that BePI provides near linear scalability in terms of preprocessing and query cost.

3.5.5 Effects of Sparse Schur Complement and Preconditioning

3.5.5.1 Effects on Preprocessing Phase

We examine the effects of the sparsification of the Schur complement (Section 3.3.4) and the preconditioning (Section 3.3.5) in the preprocessing phase of BePI. We measure the preprocessing time and the space for preprocessed data required by BePI, BePI-S, and BePI-B for each dataset.

To investigate the effect of the sparsification of the Schur complement, we first compare BePI-B with BePI-S in terms of the preprocessing time and the memory space. For preprocessing time, Figure 3.4(a) shows that BePI-S is up to 10× faster than BePI-B. For memory space, Figure 3.4(b) presents that BePI-S requires up to 5× less memory space than BePI-B. Table 3.3 summarizes the reduction of non-zero entries of the Schur complement after applying the sparsification of the Schur complement. For all datasets, the number of non-zero entries of $\mathbf{S}$ decreases by the sparsification.

Table 3.3: **Number of non-zeros of S** computed by our methods. Note that the number of non-zeros of **S** decreases by the sparsification of the Schur complement. BePI-B runs out of time (more than 24 hours) when computing **S** for the Friendster dataset, while BePI-S and BePI successfully compute it.

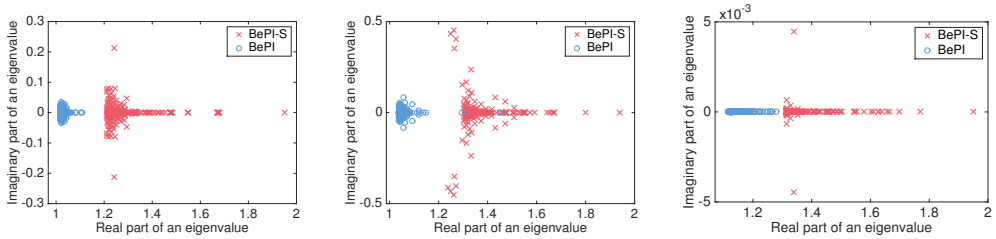
dataset	A: (|S| in BePI-B)	B: (|S| in BePI or BePI-S)	ratio (A/B)
Slashdot	664,686	353,559	1.9×
Wikipedia	844,983	626,887	1.3×
Baidu	23,136,773	2,359,563	9.8×
Flickr	113,842,305	29,990,289	3.8×
LiveJournal	417,551,300	83,070,865	5.0×
WikiLink	555,468,477	377,197,963	1.5×
Twitter	8,494,161,448	1,640,399,051	5.2×
Friendster	o.o.t.	2,018,006,285	–

Table 3.4: **Average number of iterations to compute $\mathbf{r}_2$ by BePI-S and BePI.** After preconditioning, the number of iterations for solving the linear system of **S** decreases.

dataset	A: (# iterations in BePI-S)	B: (# iterations in BePI)	ratio (A/B)
Slashdot	43.2	6.6	6.5×
Wikipedia	52.4	13.1	4.0×
Baidu	42.6	14.9	2.9×
Flickr	44.2	11.3	3.9×
LiveJournal	49.1	16.2	3.0×
WikiLink	70.2	16.5	4.3×
Twitter	60.3	18.7	3.2×
Friendster	24.2	10.5	2.3×

Especially, BePI-S reduces the number of non-zeros of **S** by 9.8× than BePI-B for the Baidu dataset. BePI-B runs out of time when computing **S** for the largest Friendster dataset.

Compared to BePI-S, BePI uses slightly more memory space as seen in Figure 3.4(b). In addition, the preprocessing phase of BePI takes slightly longer than that of BePI-S. The reason is that BePI computes the incomplete LU factors of **S**, $\tilde{\mathbf{L}}_2$ and $\tilde{\mathbf{U}}_2$, in the preprocessing phase, while BePI-S does not. However, the gap between them is small in terms of the preprocessing time and the memory space; furthermore,



(a) Slashdot (b) Wikipedia (c) Baidu

Figure 3.6: **Distribution of the top-200 eigenvalues of the preconditioned Schur complement (blue o's) and the original Schur complement (red x's).** X-axis and y-axis represent the real part and the imaginary part of an eigenvalue, respectively. Results from three different datasets, Slashdot, Wikipedia, and Baidu, show that the dispersion of eigenvalue distribution becomes much smaller when the Schur complement is preconditioned.

BePI achieves faster query time thanks to the incomplete LU factors, which we describe in the following subsection.

3.5.5.2 Effects on Query Phase

We investigate the effects of the sparsification of the Schur complement and the preconditioner on the query phase of our method. To evaluate the effects, we generate 30 random seeds, and measure the average query time using BePI, BePI-S, and BePI-B. Figure 3.4(c) compares these methods in terms of query time.

We first compare BePI-B and BePI-S to see the effect of the sparsification of the Schur complement. According to the result shown in Figure 3.4(c), BePI-S is up to 5× faster than BePI-B. This speedup is due to the reduction in the number of non-zeros of $\mathbf{S}$ by the sparsification of the Schur Complement as described in Table 3.2.

For analyzing the effect of preconditioning, we compare BePI-S and BePI. BePI is up to 4× faster than BePI-S as shown in Figure 3.4(c). Applying the preconditioner reduces the number of iterations for computing $\mathbf{r}_2$, as summarized in Table 3.4. This faster convergence is closely related to the tighter clustering of eigenvalues of the preconditioned Schur complement [53]. Figure 3.6 shows that the eigenvalues in BePI

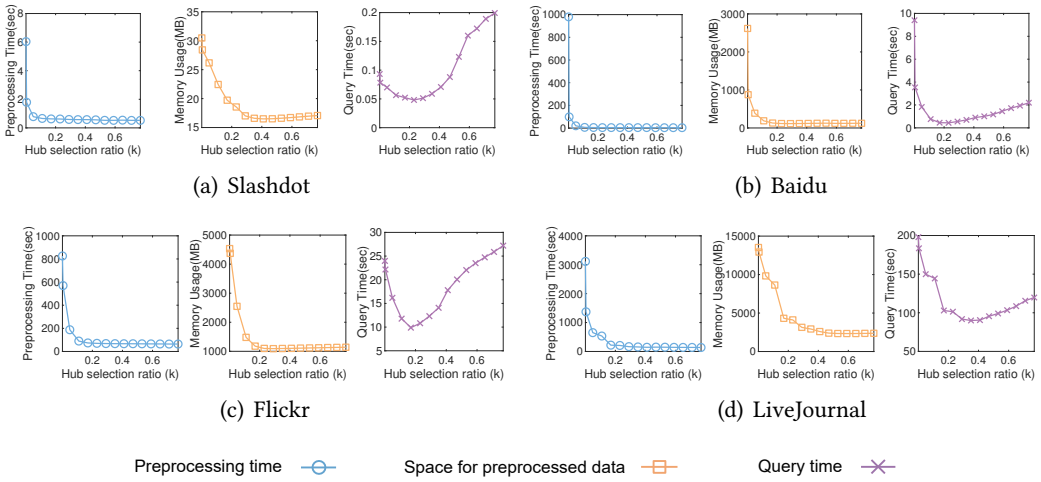


Figure 3.7: **Effects of the hub selection ratio k in Algorithm 3.** According to results, preprocessing time and memory usage of BePI decrease as k increases. Especially, when k is small (e.g., $k = 0.001$), preprocessing time and memory consumption are high. The query speed of BePI is the fastest when k is around $0.2 \sim 0.3$ as shown in the figures.

form a tight cluster, while those in BePI-S do not. In sum, BePI is up to $13\times$ faster than BePI-B in the query phase, which indicates that the query cost is effectively reduced with the sparsification of the Schur complement and the preconditioner.

3.5.6 Effects of the Hub Selection Ratio

We investigate the effects of the hub selection ratio k (Algorithm 3) on the performance of our method BePI. We measure preprocessing time, memory space of preprocessed data, and query time of BePI varying k on the Slashdot, the Baidu, the Flickr, and the LiveJournal datasets. As shown in Figure 3.7, the performance of BePI in terms of preprocessing time and memory usage becomes improved as k increases. In particular, BePI requires high preprocessing time and memory space when k is very small (e.g., $k = 0.001$). In terms of query time, BePI shows the best performance when k is from 0.2 through 0.3 as presented in Figure 3.7. There are two reasons for these effects. First, if we set a large k in Algorithm 3, then the running time of the

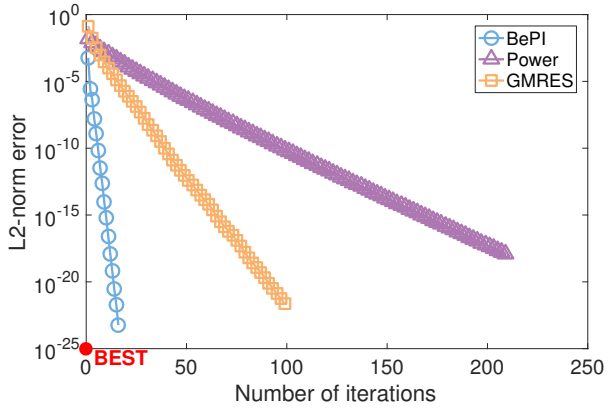


Figure 3.8: **Accuracy of BEPI** according to the number of iterations. BEPI achieves the highest accuracy and the fastest convergence compared to other iterative methods.

hub-and-spoke reordering method decreases because the number of iterations of the reordering method is reduced. Also, as described in Section 3.3.4 and Table 3.3, the number of non-zeros of the Schur complement decreases as k increases from 0; thus, the memory usage is reduced. However, setting too large k is not good for query time because the number of non-zeros and the dimension of the Schur complement become large. As shown in Figure 3.7, when k is around 0.2, it provides a good trade-off between preprocessing time, memory usage, and query time.

3.5.7 Accuracy

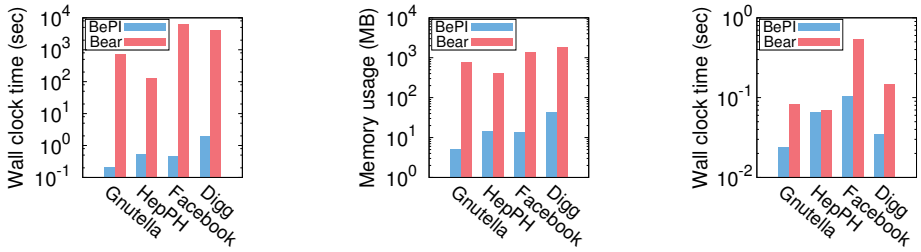
We investigate the accuracy of each iterative method compared to exact RWR solutions $\mathbf{r}^* = c\mathbf{H}^{-1}\mathbf{q}$. We perform this experiment on a small social network, the Physicians dataset [10], with 241 nodes and 1,098 edges in order to compute $\mathbf{H}^{-1}$. We select 100 seed nodes randomly, and measure average L2-norm errors between exact RWR solutions $\mathbf{r}^*$ and results $\mathbf{r}^{(i)}$ from each method with $\varepsilon = 10^{-9}$ after i-th iterations (i.e., the errors are measured by computing $\|\mathbf{r}^* - \mathbf{r}^{(i)}\|_2$). As seen in Figure 3.8, our method BEPI shows the best performance in terms of accuracy compared to other iterative

[10] **http://moreno.ss.uci.edu/data.html#ckm**

Table 3.5: **Statistics of the datasets used in Section 3.5.8.**

Dataset	Node	Edge	Description
Gnutella[1]	62,586	147,892	Peer-to-peer network
HepPH[1]	34,546	421,578	Coauthorship network
Facebook[1]	46,952	876,993	Social network
Digg[1]	279,630	1,731,653	Social network

[1] http://konect.uni-koblenz.de/



(a) Preprocessing time (b) Space for preprocessed data (c) Query time

Figure 3.9: **Detailed comparison between BEPI and Bear.** Our method BEPI significantly outperforms Bear, the state-of-the-art preprocessing method [54], in terms of preprocessing time and memory usage as shown in (a) and (b), and shows faster query speed as in (c).

methods. Furthermore, BEPI converges rapidly with higher accuracy, while power iteration and GMRES converge slowly. Note that BEPI is an exact method which can make the error smaller than any given error tolerance. As shown in Figure 3.8, the error of our method monotonically decreases and finally becomes smaller than the given error tolerance, which is also the property of the iterative method that we exploit [53].

3.5.8 Comparison with the-State-of-the-Art Method

We compare our method with Bear, the-state-of-the-art preprocessing method [54]. Since Bear suffers from the scalability issue in very large graphs as described in Section 3.5, we perform this experiment on relatively small graphs that Bear performs the preprocessing phase successfully. The datasets used in this experiments are summarized in Table 3.5. As shown in Figure 3.9, BEPI significantly outperforms Bear in

terms of preprocessing time, memory usage, and query time.

3.6 Summary

In this work, we propose BePI, a fast, memory-efficient, and scalable algorithm for random walk with restart computation on billion-scale graphs. BePI takes the advantages of both preprocessing methods and iterative methods by incorporating an iterative method within a block elimination approach. Furthermore, BePI improves the performance by decreasing the number of non-zeros of a matrix and applying a preconditioner. Consequently, BePI achieves a better scalability as well as faster query time than existing methods. We give theoretical analysis on the accuracy and complexities of BePI. Also, we experimentally show that BePI processes up to 100× larger graph, and requires up to 130× less memory space than other preprocessing methods. In the query phase, BePI computes RWR scores 9× faster than other existing methods in large graphs which other preprocessing methods fail to process, due to running out of memory or time.

Chapter 4

Personalized Ranking in Signed Graphs

4.1 Introduction

How can we obtain personalized rankings for users in signed social networks? Many social networks have allowed users to express their trust or distrust to other users. For example, in online social networks such as Slashdot [63], a user is explicitly able to mark other users as friends or foes. The users are represented as nodes, and the expressions are represented as positive and negative edges in graphs which are called *signed networks* [106]. Ranking nodes in signed networks has received much interest from data mining community to reveal trust and distrust between users [63] inducing many useful applications such as link prediction [89], anomaly detection [63], sign prediction [90], and community detection [107] in signed networks.

Traditional ranking models, however, do not provide satisfactory node rankings in signed networks. Existing random walk based ranking models such as PageRank [13] and Random Walk with Restart [37, 54, 76, 80, 108, 109] assume only positive edges; thus, they are inappropriate in the signed networks containing negative edges. Many researchers have proposed heuristics on the classical methods to make them computable in signed networks [63, 65]. However, those heuristic methods still have room to improve in terms of ranking quality since they do not consider complex social relationships such as friend-of-enemy or enemy-of-friend in their rankings as shown in Figure 4.2. In addition, most existing ranking models in signed networks

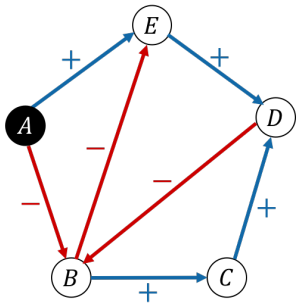
focus only on a global node ranking, although personalized rankings are more desirable for individuals in many contexts such as recommendation. Also, the fast ranking computation is important for the computational performance of applications.

In this work, we propose Signed Random Walk with Restart (SRWR), a novel model for effective personalized node rankings in signed networks. The main idea of SRWR is to introduce a sign into a random surfer in order to let the surfer consider negative edges based on structural balance theory [74, 90]. Consequently, our model considers complex edge relationships, and makes random walks interpretable in signed networks. We devise SRWR-Iter, an iterative method which naturally follows the definition of SRWR, and iteratively update SRWR scores until convergence. Furthermore, we propose SRWR-Pre, a preprocessing method for computing SRWR scores quickly which is useful for various applications in signed networks. Through extensive experiments, we demonstrate that our proposed approach offers improved performance for personalized rankings compared to alternative methods in signed social networks. Our main contributions are as follows:

- **Novel ranking model.** We propose Signed Random Walk with Restart (SRWR), a novel model for personalized rankings in signed networks (Definition 4.1). We show that our model is a generalized version of RWR working on both signed and unsigned networks (Property 3).
- **Algorithm.** We propose SRWR-Iter and SRWR-Pre for computing SRWR scores. SRWR-Iter is an iterative algorithm which naturally follows the definition of SRWR (Algorithm 6). SRWR-Pre is a preprocessing method which employs a node reordering technique and block elimination to accelerate SRWR computation speed (Algorithms 7 and 8).
- **Experiment.** We show that SRWR achieves higher accuracy for link predic-

Table 4.1: **Table of symbols used in Chapter 4.** Boldface capital letters, such as $\mathbf{A}$, represent matrices. Boldface small letters, such as $\mathbf{r}$, represent vectors.

Symbol	Definition
$G = (\mathbf{V}, \mathbf{E})$	signed input graph
$\mathbf{V}$	set of nodes in G
$\mathbf{E}$	set of signed edges in G
n	number of nodes in G
n_1	number of spokes in G
n_2	number of hubs in G
m	number of edges in G
s	seed node (= query node, source node)
c	restart probability
ε	error tolerance
$\overleftarrow{\mathbf{N}}_u$	set of in-neighbors to nodes u
$\overrightarrow{\mathbf{N}}_u$	set of out-neighbors from nodes u
$\mathbf{A}$	$(n \times n)$ signed adjacency matrix of G
$\lvert\mathbf{A}\rvert$	$(n \times n)$ absolute adjacency matrix of G
$\mathbf{D}$	$(n \times n)$ out-degree matrix of $\lvert\mathbf{A}\rvert$, $\mathbf{D}_{ii} = \sum_j \lvert\mathbf{A}\rvert_{ij}$
$\tilde{\mathbf{A}}$	$(n \times n)$ semi-row normalized matrix of $\mathbf{A}$
$\tilde{\mathbf{A}}_+$	$(n \times n)$ positive semi-row normalized matrix of $\mathbf{A}$
$\tilde{\mathbf{A}}_-$	$(n \times n)$ negative semi-row normalized matrix of $\mathbf{A}$
$\lvert\tilde{\mathbf{A}}\rvert$	$(n \times n)$ absolute row-normalized matrix of $\lvert\mathbf{A}\rvert$
$\mathbf{q}$	$(n \times 1)$ starting vector (= s-th unit vector)
$\mathbf{r}^+$	$(n \times 1)$ positive score vector
$\mathbf{r}^-$	$(n \times 1)$ negative score vector
$\mathbf{r}$	$(n \times 1)$ trustworthiness score vector, e.g., $\mathbf{r} = \mathbf{r}^+ - \mathbf{r}^-$
$\mathbf{p}$	$(n \times 1)$ $\mathbf{p} = \mathbf{r}^+ + \mathbf{r}^-$
$\lvert\mathbf{H}\rvert$	$(n \times n)$ $\lvert\mathbf{H}\rvert = \mathbf{I} - (1-c)\lvert\tilde{\mathbf{A}}\rvert^\top$
$\mathbf{T}$	$(n \times n)$ $\mathbf{T} = \mathbf{I} - (1-c)(\gamma\tilde{\mathbf{A}}_+^\top - \beta\tilde{\mathbf{A}}_-^\top)$
$\lvert\mathbf{H}\rvert_{ij}, \mathbf{T}_{ij}$	$(n_i \times n_j)$ (i,j)-th partition of $\lvert\mathbf{H}\rvert$ or $\mathbf{T}$
$\mathbf{S}_{\lvert\mathbf{H}\rvert}$, $\mathbf{S}_\mathbf{T}$	$(n_2 \times n_2)$ Schur complement of $\lvert\mathbf{H}\rvert_{11}$ or $\mathbf{T}_{11}$
$\mathbf{q}_i$, $\mathbf{p}_i$, $\mathbf{r}_i^-$	$(n_i \times 1)$ i-th partition of $\mathbf{q}$, $\mathbf{p}$ or $\mathbf{r}^-$



Rank	Node	r: Trust-worthiness	r^+: Positive score	r^-: Negative score
1st	A	0.2500	0.2500	0.0000
2nd	E	0.1487	0.1687	0.0200
3rd	D	0.0703	0.1416	0.0713
4th	C	-0.0549	0.0200	0.0750
5th	B	-0.1465	0.0534	0.1999

Input: a signed network & seed node A

Output: the trustworthiness score vector $\boldsymbol{r}$ w.r.t. the seed node

Figure 4.1: **Example of the personalized node ranking problem in Problem 1.** Given a signed network and a seed node (in this example, node A is the seed node), our goal is to compute the trustworthiness score vector $\mathbf{r}$ w.r.t. the seed node. Our proposed model SRWR (see Definition 4.1 in Section 4.3) aims to compute $\mathbf{r}$ based on the positive and negative score vectors $\mathbf{r}^+$ and $\mathbf{r}^-$, i.e., $\mathbf{r} = \mathbf{r}^+ - \mathbf{r}^-$.

tion (Figure 4.7), predicts trolls 4× more accurately (Figure 4.9), and provides a good performance for sign prediction compared to other ranking models (Figure 4.10). In terms of efficiency, SRWR-Pre preprocesses signed networks up to 4.5× faster, and requires 11× less memory space than baseline preprocessing methods. Furthermore, SRWR-Pre computes SRWR scores up to 14× faster than other methods including SRWR-Iter (Figure 4.13).

The code of our method and datasets used in this paper are available at **http://datalab.snu.ac.kr/srwrpre**. The rest of this paper is organized as follows. We first introduce the formal definition of the personalized ranking problem in signed networks at Section 4.2. In Section 4.3, we describe our proposed model and algorithms for computing personalized rankings. After presenting experimental results in Section 4.4, we summarize this work in Section 4.5. Table 4.1 lists the symbols used in this chapter.

4.2 Problem Definition

We define the personalized ranking problem in signed networks as follows:

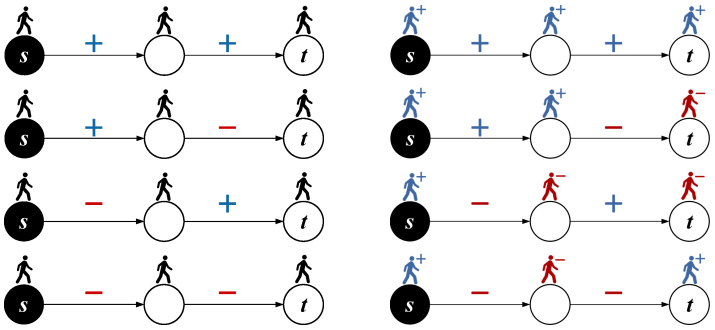
Problem 1 (Personalized Node Ranking in Signed Networks)**.**

- ***Input:*** *a signed network $G = (\boldsymbol{V},\boldsymbol{E})$ and a seed node s where $\boldsymbol{V}$ is the set of nodes, and $\boldsymbol{E}$ is the set of signed edges.*
- ***Output:*** *a trustworthiness score vector $\mathbf{r} \in \mathbb{R}^n$ of all other nodes for seed node s to rank those nodes w.r.t. seed node s.* ■

In signed social networks, users are represented as nodes, and trust or distrust relations between users are represented as positive or negative edges. When a user u considers that a user v is trustworthy, a positive edge $u \rightarrow v$ is formed. On the contrary, a negative edge $u \rightarrow v$ is formed when u distrusts v. Given those signed edges between nodes and a seed node s, the personalized ranking problem is to rank all other nodes w.r.t. seed node s in the order of trustworthiness scores represented by $\mathbf{r}$ where $\mathbf{r}_u$ indicates how much seed node s should trust node u as depicted in Figure 4.1. If the score $\mathbf{r}_u$ is high, then s is likely to trust u. Otherwise, s is likely to distrust u.

4.3 Proposed Method

We propose Signed Random Walk with Restart (SRWR), a novel ranking model for signed networks in Section 4.3.1. Then we first develop an iterative algorithm SRWR-Iter for computing SRWR scores w.r.t. a seed node in Section 4.3.2, and then propose a preprocessing algorithm SRWR-Pre to accelerate SRWR computation speed in Section 4.3.3.



(a) Traditional random walks (b) Signed random walks

Figure 4.2: **Examples of traditional random walks and signed random walks.** Each case represents 1) friend's friend, 2) friend's enemy, 3) enemy's friend, or 4) enemy's enemy from the top. A random surfer has either a positive (blue) or a negative (red) sign on each node in Figure 4.2(b). When the signed surfer traverses a negative edge, she changes her sign from positive to negative or vice versa.

4.3.1 Signed Random Walk with Restart Model

As discussed in Section 4.1, complicated relationships of signed edges are the main obstacles for providing effective rankings in signed networks. Most existing works on signed networks have not focused on personalized rankings. In this work, our goal is to design a novel ranking model which resolves those problems in signed networks. The main ideas of our model are as follows:

- We introduce a signed random surfer. The sign of the surfer is either positive or negative, which means favorable or adversarial to a node, respectively.
- When the random surfer encounters a negative edge, she changes her sign from positive to negative, or vice versa. Otherwise, she keeps her sign.
- We introduce balance attenuation factors into the surfer to consider the uncertainty for friendship of enemies.

There are four cases according to the signs of edges as shown in Figure 4.2: 1) friend's friend, 2) friend's enemy, 3) enemy's friend, and 4) enemy's enemy. Suppose a random surfer starts at node s toward node t. A traditional surfer just moves along

the edges without considering signs as seen in Figure 4.2(a) since there is no way to consider the signs on the edges. Hence, classical models cannot distinguish those edge relationships during her walks. For instance, the model considers that node s and node t are friends for the second case (friend's enemy), even though node t are more likely to be an enemy w.r.t. node s.

On the contrary, our model in Figure 4.2(b) has a signed random surfer who considers those complex edge relationships. If the random surfer starting at node s with a positive sign encounters a negative edge, she flips her sign from positive to negative, or vice versa. Our model distinguishes whether node t is the friend of node s or not according to her sign at node t. As shown in Figure 4.2(b), the results for all cases from our model are consistent with structural balance theory [74]. Thus, introducing a signed random surfer enables our model to discriminate those edge relationships.

Trust or distrust relationships between a specific node s and other nodes are revealed as the surfer is allowed to move around a signed network starting from node s. If the positive surfer visits a certain node u many times, then node u is trustable for node s. On the other hand, if the negative surfer visits node u many times, then node s is not likely to trust node u. Thus, rankings are obtained by revealing a degree of trust or distrust between people based on the signed random walks. Here, we formally define our model on signed networks in Definition 4.1. Note that Definition 4.1 involves the concept of restart which provides personalized rankings w.r.t. a user.

Definition 4.1 (Signed Random Walk with Restart)**.** *A signed random surfer has a sign, which is either positive or negative. At the beginning, the surfer starts with $+$ sign from a seed node s because she trusts s. Suppose the surfer is currently at node u, and c is the restart probability of the surfer. Then, she takes one of the following actions:*

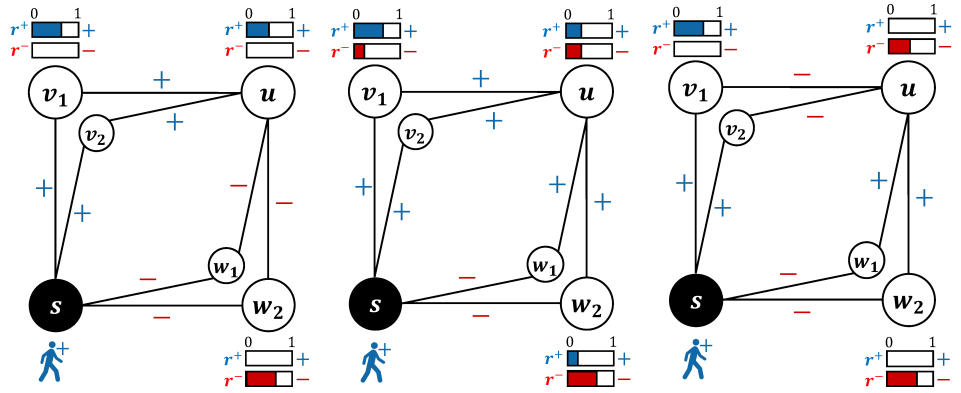
- ***Action 1: Signed Random Walk.*** *The surfer randomly moves to one of the neighbors from node u with probability $1-c$. The surfer flips her sign if she encounters a negative edge. Otherwise, she keeps her sign.*
- ***Action 2: Restart.*** *The surfer goes back to the seed node s with probability c. Her sign should become $+$ at the seed node s because she trusts s.* ■

We measure two probabilities on each node through Signed Random Walk with Restart (SRWR) starting from the seed node s. The two probabilities are represented as follows:

- $\mathbf{r}_u^+ = P(u,+)$: the probability that the positive surfer visits node u after SRWR from seed node s.
- $\mathbf{r}_u^- = P(u,-)$: the probability that the negative surfer visits node u after SRWR from seed node s.

Note that $\mathbf{r}_u^+$ (or $\mathbf{r}_u^-$) corresponds to a ratio of how many times the positive (or negative) surfer visits node u during SRWR. If the positive surfer visits node u much more than the negative one, then s is likely to trust u. Otherwise, s is likely to distrust u. In other words, s would consider u as a positive node if $\mathbf{r}_u^+$ is greater than $\mathbf{r}_u^-$. On the contrary, s would treat u as a negative one if $\mathbf{r}_u^-$ is greater than $\mathbf{r}_u^+$. Based on this intuition, we define the relative trustworthiness score $\mathbf{r}_u = \mathbf{r}_u^+ - \mathbf{r}_u^-$ between s and u. For all nodes, $\mathbf{r}^+$ is a positive score vector and $\mathbf{r}^-$ is a negative score vector of SRWR. Then, the trustworthiness score vector for SRWR is represented as $\mathbf{r} = \mathbf{r}^+ - \mathbf{r}^-$, the output of Problem 1. Many researchers have dealt with trust and distrust between nodes through such representation for trustworthiness [63, 65, 110, 91]. Especially, the interpretation of the resulting values from $\mathbf{r}_u = \mathbf{r}_u^+ - \mathbf{r}_u^-$ is consistent with what Kunegis et al. said as follows:

- *"The resulting popularity (based on trustworthiness) measure admits both posi-*



(a) A trustful case (b) A neutral case (c) A distrustful case

Figure 4.3: **Examples of how to interpret positive and negative scores of SRWR** between nodes s and u. The bars on node u depict how many the signed surfer visits that node, indicating positive and negative scores between s and u. (a) and (c) represent trustful and distrustful cases between those nodes: s is likely to trust u in (a), and s is likely to distrust u in (c). However, if the those scores are similar as in (b), it is difficult for node s to decide whether to trust node u or not. Hence, s is likely to be neutral about node u in (b).

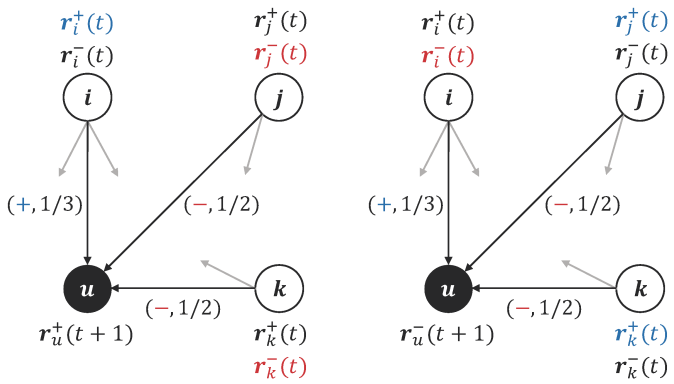
tive and negative values, and represents a measure of popularity in the network, with positive edges corresponding to a positive endorsement and negative edges to negative endorsements. This interpretation is consistent with the semantics of the 'friend' and 'foe' relationships [63]."

Note that from the viewpoint of measure theory, the relative trustworthiness $\mathbf{r}_u$ is also an acceptable measure as *signed measure* [111] if we consider $\mathbf{r}_u^+$ and $\mathbf{r}_u^-$ as non-negative measures (i.e., $\mathbf{r}_u^+ \geq 0$ and $\mathbf{r}_u^- \geq 0$). We discuss this in detail in Appendix A.5.

Discussion on positive and negative SRWR scores. We explain how to interpret positive and negative SRWR scores using an example in Figure 4.3. Suppose the signed surfer starts at node s, and performs SRWR to measure the trustworthiness between nodes s and u. Note that the trustworthiness score depends on which signed surfer stays at node u more frequently. Then, there would be three cases depending on the link structure between s and u as shown in Figure 4.3. For the case in Figure 4.3(a), s is likely to trust u since the positive surfer visits u much more than

the negative surfer through paths from s to u (i.e., the positive score is larger than the negative one at u). For the opposite case in Figure 4.3(c), s is likely to distrust u because the negative surfer frequently visits u. However, if those scores on u are similar as shown in Figure 4.3(b), then it is hard for s to determine whether to trust u or not. In this case s is likely to be neutral about node u. Thus, the trustworthiness score $\mathbf{r}_u$ of the trustful case is high (and positive in SRWR), and that of the distrustful case is low (and negative in SRWR). For the neutral case, the score would be in the middle (and around zero between -1 and 1 in SRWR).

Connection to balance theory. According to balance theory [74, 112], Figure 4.3(a) and 4.3(c) are balanced networks because the graphs are divided into two sets of users with mutual antagonism between the sets. For example, the set of nodes $\{v_1, v_2, s\}$ and the other set of nodes $\{w_1, w_2, u\}$ in Figure 4.3(c) are connected with negative edges, and nodes in each set are positively connected. In the balanced networks, each node has either a positive score or a negative one. Because the signed surfer changes her sign walking negative edges linking the two groups, the positive surfer stays and walks only in one group and the negative surfer stays and walks only in the other group. However, Figure 4.3(b) is an unbalanced network because it cannot be divided into two sets that are negatively connected each other. Hence, positive and negative surfers visits the same node, i.e., each node has both positive and negative scores. In this case, the trustworthiness score on a node is determined by which signed surfer visits the node more frequently, which is represented by the difference between positive and negative scores.



(a) An example of a positive probability, $\mathbf{r}_u^+(t+1)$

(b) An example of a negative probability, $\mathbf{r}_u^-(t+1)$

Figure 4.4: **Examples of how $\mathbf{r}_u^+$ and $\mathbf{r}_u^-$ are defined in SRWR.**

4.3.1.1 Formulation for Signed Random Walk with Restart

We formulate the probability vectors, $\mathbf{r}^+$ and $\mathbf{r}^-$, following Signed Random Walk with Restart. First, we explain how to define $\mathbf{r}^+$ and $\mathbf{r}^-$ using the example shown in Figure 4.4. In the example, we label a (sign, transition probability) pair on each edge. For instance, the transition probability for the positive edge from node i to node u is $1/3$ because node i has 3 outgoing edges. This edge is denoted by $(+, 1/3)$. Other pairs of signs and transition probabilities are also similarly defined. In order that the random surfer has a positive sign on node u at time $t+1$, a positive surfer on one of u's neighbor at time t must move to node u through a positive edge, or a negative surfer must move through a negative edge according to the signed random walk action in Definition 4.1. Considering the restart action of the surfer with the probability c, $\mathbf{r}_u^+(t+1)$ in Figure 4.4(a) is represented as follows:

$$\mathbf{r}_u^+(t+1) = (1-c)\left(\frac{\mathbf{r}_i^+(t)}{3} + \frac{\mathbf{r}_j^-(t)}{2} + \frac{\mathbf{r}_k^-(t)}{2}\right) + c\mathbf{1}(u=s)$$

where $\mathbf{1}(u=s)$ is 1 if u is the seed node s and 0 otherwise. In Figure 4.4(b), $\mathbf{r}_u^-(t+1)$ is defined similarly as follows:

$$\mathbf{r}_u^-(t+1) = (1-c)\left(\frac{\mathbf{r}_i^-(t)}{3} + \frac{\mathbf{r}_j^+(t)}{2} + \frac{\mathbf{r}_k^+(t)}{2}\right)$$

Note that we do not add the restarting score $c\mathbf{1}(u=s)$ to $\mathbf{r}_u^-(t+1)$ in this case because the surfer's sign must become positive when she goes back to the seed node s. The recursive equations of our model are defined as follows:

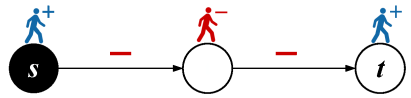
$$\begin{aligned}
\mathbf{r}_u^+ &= (1-c)\left(\sum_{v\in\overleftarrow{\mathbf{N}}_u^+}\frac{\mathbf{r}_v^+}{|\overrightarrow{\mathbf{N}}_v|} + \sum_{v\in\overleftarrow{\mathbf{N}}_u^-}\frac{\mathbf{r}_v^-}{|\overrightarrow{\mathbf{N}}_v|}\right) + c\mathbf{1}(u=s) \\
\mathbf{r}_u^- &= (1-c)\left(\sum_{v\in\overleftarrow{\mathbf{N}}_u^-}\frac{\mathbf{r}_v^+}{|\overrightarrow{\mathbf{N}}_v|} + \sum_{v\in\overleftarrow{\mathbf{N}}_u^+}\frac{\mathbf{r}_v^-}{|\overrightarrow{\mathbf{N}}_v|}\right)
\end{aligned} \tag{4.1}$$

where $\overleftarrow{\mathbf{N}}_i$ is the set of in-neighbors of node i, and $\overrightarrow{\mathbf{N}}_i$ is the set of out-neighbors of node i. Superscripts of $\overleftarrow{\mathbf{N}}_i$ or $\overrightarrow{\mathbf{N}}_i$ indicate signs of edges between node i and its neighbors (e.g., $\overleftarrow{\mathbf{N}}_i^+$ indicates the set of positively connected in-neighbors of node i). We need to introduce several symbols related to an adjacency matrix $\mathbf{A}$ to vectorize Equation (4.1).

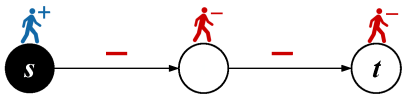
Definition 4.2 (Signed adjacency matrix)**.** *The signed adjacency matrix* $\mathbf{A}$ *of* G *is a matrix such that* $\mathbf{A}_{uv}$ *is positive or negative when there is a positive or a negative edge from node* u *to node* v *respectively, and zero otherwise.* ■

Definition 4.3 (Semi-row normalized matrix)**.** *Let* $|\mathbf{A}|$ *be the absolute adjacency matrix of* $\mathbf{A}$*, and* $\mathbf{D}$ *be the out-degree diagonal matrix of* $|\mathbf{A}|$ *(i.e.,* $\mathbf{D}_{ii} = \sum_j |\mathbf{A}|_{ij}$*). Then semi-row normalized matrix of* $\mathbf{A}$ *is* $\tilde{\mathbf{A}} = \mathbf{D}^{-1}\mathbf{A}$. ■

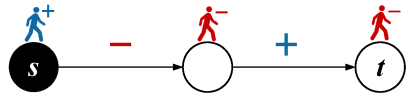
Definition 4.4 (Positive or negative semi-row normalized matrix)**.** *The positive semi-*



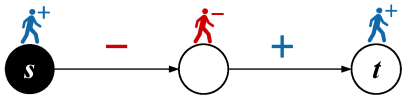
(a) The surfer's sign at node t is positive with β (balanced case)



(b) The surfer's sign at node t is negative with $1-\beta$ (unbalanced case)



(c) The surfer's sign at node t is negative with γ (balanced case)



(d) The surfer's sign at node t is positive with $1-\gamma$ (unbalanced case)

Figure 4.5: **Examples of balance attenuation factors.** (a) and (b) represent the uncertainty for "the enemy of my enemy is my friend" with probability β, and (c) and (d) represent the uncertainty for "the friend of my enemy is my enemy" with probability γ.

row normalized matrix $\tilde{\mathbf{A}}_+$ *contains only positive values in the semi-row normalized matrix* $\tilde{\mathbf{A}}$*. The negative semi-row normalized matrix* $\tilde{\mathbf{A}}_-$ *contains absolute values of negative elements in* $\tilde{\mathbf{A}}$*. In other words,* $\tilde{\mathbf{A}} = \tilde{\mathbf{A}}_+ - \tilde{\mathbf{A}}_-$*, and* $|\tilde{\mathbf{A}}| = \tilde{\mathbf{A}}_+ + \tilde{\mathbf{A}}_-$*.* ■

Based on Definitions 4.3 and 4.4, Equation (4.1) is represented as follows:

$$\begin{aligned} \mathbf{r}^+ &= (1-c)\left(\tilde{\mathbf{A}}_+^\top \mathbf{r}^+ + \tilde{\mathbf{A}}_-^\top \mathbf{r}^-\right) + c\mathbf{q} \\ \mathbf{r}^- &= (1-c)\left(\tilde{\mathbf{A}}_-^\top \mathbf{r}^+ + \tilde{\mathbf{A}}_+^\top \mathbf{r}^-\right) \end{aligned} \tag{4.2}$$

where $\mathbf{q}$ is a vector whose sth element is 1 and all other elements are 0.

4.3.1.2 Balance Attenuation Factors

The signed surfer measures positive and negative scores of nodes w.r.t. a seed node in terms of trust and distrust according to edge relationships as discussed in Section 4.3.1. Our model in Definition 4.1 strongly supports the four cases between nodes in Figure 4.2(b) where those cases represent *strong balance theory* [75, 74]. However, recent works [113] have argued that the strong balance theory is unsatisfactory

for fully supporting real-world signed networks, since unbalanced relationships frequently appear. Thus, this limitation would be naturally inherent in our model. To alleviate this limitation, many researchers have studied *weak balance theory* [114, 113] which generalizes the strong balance theory by allowing several unbalanced cases such as "the enemy of my enemy is my enemy". Similarly, we adopt the generalization strategy of the weak balance theory to make our model flexible on unbalanced networks through dealing with both balanced and unbalanced cases.

We consider that the relationship of enemies of a seed user is uncertain since the user cannot believe the information provided by her enemies. We reflect the uncertainty of the relationship of those enemies into our ranking model by introducing stochastic parameters, β and γ, called *balance attenuation factors*. Note that we assume that the positive and negative relationship of friends of the seed user is reliable since the user trusts her friends. β is a parameter for the uncertainty of "the enemy of my enemy is my friend", and γ is for "the friend of my enemy is my enemy." We first explain β using the fourth case (enemy's enemy) in Figure 4.2(b). Suppose a surfer with a positive sign starts at node s toward node t and encounters two consecutive negative edges. Based on strong balance theory, her sign becomes negative at the intermediate node m and positive at node t in Figure 4.5(a). However, some people might think that the enemy of my enemy is my enemy as shown in Figure 4.5(b). In this case, her sign will be negative at nodes m and t. To consider this uncertainty, we introduce a parameter β so that if the negative surfer at node m encounters a negative edge, her sign becomes positive with probability β or negative with $1-\beta$ at node t. The other parameter γ is also interpreted similarly to β. When the negative surfer at node m encounters a positive edge, her sign will be negative with probability γ or positive with $1-\gamma$ at node t as in Figures 4.5(c) and 4.5(d). SRWR with the balance

attenuation factors is represented as follows:

$$\begin{aligned}\mathbf{r}^{+} &= (1-c)\left(\tilde{\mathbf{A}}_{+}^{\top}\mathbf{r}^{+} + \beta\tilde{\mathbf{A}}_{-}^{\top}\mathbf{r}^{-} + (1-\gamma)\tilde{\mathbf{A}}_{+}^{\top}\mathbf{r}^{-}\right) + c\mathbf{q} \\ \mathbf{r}^{-} &= (1-c)\left(\tilde{\mathbf{A}}_{-}^{\top}\mathbf{r}^{+} + \gamma\tilde{\mathbf{A}}_{+}^{\top}\mathbf{r}^{-} + (1-\beta)\tilde{\mathbf{A}}_{-}^{\top}\mathbf{r}^{-}\right)\end{aligned} \tag{4.3}$$

Discussion on other balance attenuation factors. Note that other parameters for the uncertainties of "enemy of friend" and "friend of friend" could be easily adopted into our model. However, we do not reflect those parameters on our model with the following reasons:

- As described in this subsection, we assume that the positive and negative relationship of friends of a seed user is reliable and stable. If the seed user's friends distrust a user, then she is unlikely to believe the user since the user trusts her friends.
- Introducing the additional parameters could improve the performance of applications in signed networks, but it increases the complexity of our model considering too many uncertain cases. We consider that introducing β and γ achieves a good trade-off between the model complexity and the performance of each application as shown in Section 4.4.

Discussion on the initial sign. In Definition 4.1, we initialize the signed surfer as positive when she restarts at a seed node s. One might consider that our model is easily extendable to probabilistically initializing the signed surfer as negative for the restart action. Let p denote the probability of being the positive surfer for the restart action. Then, the extended version is established by changing $c\mathbf{q}$ to $(c \times p)\mathbf{q}$ in the first equation and adding $(c \times (1-p))\mathbf{q}$ into the second equation of equation (4.3). However, we do not consider such case with the following reason:

- If the negative surfer starts at s, the surfer becomes positive at nodes negatively

connected from s and negative at those positively connected from s. This implies that the surfer recognizes the friends of s as enemies and the enemies of s as friends. Thus, it is hard to interpret the scores measured by the negative initial surfer in terms of trustworthiness for s based on balance theory.

4.3.2 SRWR-Iter: Iterative Algorithm for Signed Random Walk with Restart

We present an iterative algorithm SRWR-Iter for computing SRWR scores based on Equation (4.3). Note that the solution of a linear system with recursive structure is typically and efficiently obtained via an iterative manner such as power iteration and Jacobi method [115]. We also adopt such iterative strategy to solve the recursive equations in Equation (4.3). We describe how SRWR-Iter obtains the trustworthiness SRWR score vector $\mathbf{r}$ given a signed network and a seed node in Algorithms 5 and 6. Moreover, we prove that the iterative approach in SRWR-Iter converges, and returns a unique solution for the seed node in Theorem 4.1 of Section 4.3.2.2.

Normalization phase (Algorithm 5). Our proposed algorithm first computes the out-degree diagonal matrix $\mathbf{D}$ of $|\mathbf{A}|$, which is the absolute adjacency matrix of $\mathbf{A}$ (line 1). Then, the algorithm computes the semi-row normalized matrix $\tilde{\mathbf{A}}$ using $\mathbf{D}$ (line 2). We split $\tilde{\mathbf{A}}$ into two matrices: the positive semi-row normalized matrix ($\tilde{\mathbf{A}}_+$) and the negative semi-row normalized matrix ($\tilde{\mathbf{A}}_-$) (line 3) satisfying $\tilde{\mathbf{A}} = \tilde{\mathbf{A}}_+ - \tilde{\mathbf{A}}_-$.

Iteration phase (Algorithm 6). Our algorithm computes the SRWR score vectors $\mathbf{r}^+$ and $\mathbf{r}^-$ for the seed node s with the balance attenuation factors (β and γ) in the iteration phase. We set $\mathbf{q}$ to s-th unit vector, and initialize $\mathbf{r}^+$ to $\mathbf{q}$ and $\mathbf{r}^-$ to $\mathbf{0}$ (lines 1 and 2). Our algorithm iteratively computes Equation (4.3) (lines 4 and 5). We concatenate $\mathbf{r}^+$ and $\mathbf{r}^-$ vertically (line 6) into $\mathbf{h}$. We then compute the error δ between

Algorithm 5: Normalization phase of SRWR-ITER

Input: signed adjacency matrix: $\mathbf{A}$
Output: positive semi-row normalized matrix: $\tilde{\mathbf{A}}_+$, and negative semi-row normalized matrix: $\tilde{\mathbf{A}}_-$

1: compute out-degree matrix $\mathbf{D}$ of $|\mathbf{A}|$, $\mathbf{D}_{ii} = \sum_j |\mathbf{A}|_{ij}$
2: compute semi-row normalized matrix, $\tilde{\mathbf{A}} = \mathbf{D}^{-1}\mathbf{A}$.
3: split $\tilde{\mathbf{A}}$ into $\tilde{\mathbf{A}}_+$ and $\tilde{\mathbf{A}}_-$ such that $\tilde{\mathbf{A}} = \tilde{\mathbf{A}}_+ - \tilde{\mathbf{A}}_-$
4: **return** $\tilde{\mathbf{A}}_+$ and $\tilde{\mathbf{A}}_-$

$\mathbf{h}$ and $\mathbf{h}'$ which is the result in the previous iteration (line 7). We update $\mathbf{h}$ into $\mathbf{h}'$ for the next iteration (line 8). The iteration stops when the error δ is smaller than a threshold ε (line 9). We finally return the trustworthiness score vector $\mathbf{r}$ used for the personalized ranking w.r.t. s by computing $\mathbf{r} = \mathbf{r}^+ - \mathbf{r}^-$ (lines 10 and 11).

4.3.2.1 Space and Time Complexities of SRWR-ITER

Lemma 4.1 (Space and Time Complexities of SRWR-ITER). *Let n and m denote the number of nodes and edges of a signed network, respectively. Then, the space complexity of Algorithm 6 is $O(n+m)$. The time complexity of Algorithm 6 is $O(T(n+m))$ where the number T of iterations is $\log_{1-c}\frac{\varepsilon}{2}$, c is the restart probability, and ε is an error tolerance.*

Proof. The space complexity for $\tilde{\mathbf{A}}_+$ and $\tilde{\mathbf{A}}_-$ is $O(m)$ if we exploit a sparse matrix format such as compressed column storage to save the matrices. We need $O(n)$ for SRWR score vectors $\mathbf{r}^+$ and $\mathbf{r}^-$. Thus, the space complexity is $O(n+m)$. One iteration in Algorithm 6 takes $O(n+m)$ time due to sparse matrix vector multiplications and vector additions where the time complexity of a sparse matrix vector multiplication is linear to the number of non-zeros of a matrix [100]. Hence, the total time complexity is $O(T(n+m))$ where the number T of iterations is $\log_{1-c}\frac{\varepsilon}{2}$ which is proved in Lemma 4.2. □

Algorithm 6: Iteration phase of SRWR-ITER

Input: positive semi-row normalized matrix: $\tilde{\mathbf{A}}_+$, and negative semi-row normalized matrix: $\tilde{\mathbf{A}}_-$, and seed node: s, restart probability: c, balance attenuation factors: β and γ, and error tolerance: ε.

Output: trustworthiness SRWR score vector: $\mathbf{r}$

set the starting vector $\mathbf{q}$ from the seed node s
set $\mathbf{r}^+ = \mathbf{q}$, $\mathbf{r}^- = \mathbf{0}$, and $\mathbf{h}' = [\mathbf{r}^+; \mathbf{r}^-]$
repeat
  $\mathbf{r}^+ \leftarrow (1-c)(\tilde{\mathbf{A}}_+^\top \mathbf{r}^+ + \beta\tilde{\mathbf{A}}_-^\top \mathbf{r}^- + (1-\gamma)\tilde{\mathbf{A}}_+^\top \mathbf{r}^-) + c\mathbf{q}$
  $\mathbf{r}^- \leftarrow (1-c)(\tilde{\mathbf{A}}_-^\top \mathbf{r}^+ + \gamma\tilde{\mathbf{A}}_+^\top \mathbf{r}^- + (1-\beta)\tilde{\mathbf{A}}_-^\top \mathbf{r}^-)$
  concatenate $\mathbf{r}^+$ and $\mathbf{r}^-$ into $\mathbf{h} = [\mathbf{r}^+; \mathbf{r}^-]^\top$
  compute the error between $\mathbf{h}$ and $\mathbf{h}'$, $\delta = \|\mathbf{h} - \mathbf{h}'\|$
  update $\mathbf{h}' \leftarrow \mathbf{h}$ for the next iteration
until $\delta < \varepsilon$
compute $\mathbf{r} = \mathbf{r}^+ - \mathbf{r}^-$
return $\mathbf{r}$

4.3.2.2 Theoretical Analysis of Iterative Algorithm and Signed Random Walk with Restart

We theoretically analyze the iterative algorithm SRWR-ITER and the properties of Signed Random Walk with Restart.

Convergence Analysis of SRWR-ITER. We show that the iteration in Algorithm 6 converges to the solution of a linear system as described in the following theorem.

Theorem 4.1 (Convergence of SRWR-ITER). *Suppose* $\mathbf{h} = [\mathbf{r}^+; \mathbf{r}^-]^\top$ *and* $\mathbf{q}_s = [\mathbf{q}; \mathbf{0}]^\top$. *Then the iteration for* $\mathbf{h}$ *in Algorithm 6 converges to the solution* $\mathbf{h} = c(\mathbf{I} - (1-c)\tilde{\mathbf{B}}^\top)^{-1}\mathbf{q}_s$ *where* $\tilde{\mathbf{B}}^\top = \begin{bmatrix} \tilde{\mathbf{A}}_+^\top & \beta\tilde{\mathbf{A}}_-^\top + (1-\gamma)\tilde{\mathbf{A}}_+^\top \\ \tilde{\mathbf{A}}_-^\top & (1-\beta)\tilde{\mathbf{A}}_-^\top + \gamma\tilde{\mathbf{A}}_+^\top \end{bmatrix}$.

Proof. Equation (4.3) is represented as follows:

$$\begin{bmatrix} \mathbf{r}^+ \\ \mathbf{r}^- \end{bmatrix} = (1-c)\begin{bmatrix} \tilde{\mathbf{A}}_+^\top & \beta\tilde{\mathbf{A}}_-^\top + (1-\gamma)\tilde{\mathbf{A}}_+^\top \\ \tilde{\mathbf{A}}_-^\top & (1-\beta)\tilde{\mathbf{A}}_-^\top + \gamma\tilde{\mathbf{A}}_+^\top \end{bmatrix}\begin{bmatrix} \mathbf{r}^+ \\ \mathbf{r}^- \end{bmatrix} + c\begin{bmatrix} \mathbf{q} \\ \mathbf{0} \end{bmatrix} \Leftrightarrow \mathbf{h} = (1-c)\tilde{\mathbf{B}}^\top\mathbf{h} + c\mathbf{q}_s$$

where $\tilde{\mathbf{B}}^\top = \begin{bmatrix} \tilde{\mathbf{A}}_+^\top & \beta\tilde{\mathbf{A}}_-^\top + (1-\gamma)\tilde{\mathbf{A}}_+^\top \\ \tilde{\mathbf{A}}_-^\top & (1-\beta)\tilde{\mathbf{A}}_-^\top + \gamma\tilde{\mathbf{A}}_+^\top \end{bmatrix}$, $\mathbf{h} = \begin{bmatrix} \mathbf{r}^+ \\ \mathbf{r}^- \end{bmatrix}$, and $\mathbf{q}_s = \begin{bmatrix} \mathbf{q} \\ \mathbf{0} \end{bmatrix}$. Thus, the iteration in Algorithm 6 is written as in the following equation:

$$\begin{aligned} \mathbf{h}^{(k)} &= (1-c)\tilde{\mathbf{B}}^\top \mathbf{h}^{(k-1)} + c\mathbf{q}_s \\ &= \left((1-c)\tilde{\mathbf{B}}^\top\right)^2 \mathbf{h}^{(k-2)} + \left((1-c)\tilde{\mathbf{B}}^\top + \mathbf{I}\right) c\mathbf{q}_s \\ &= \cdots \\ &= \left((1-c)\tilde{\mathbf{B}}^\top\right)^k \mathbf{h}^{(0)} + \left(\sum_{j=0}^{k-1} \left((1-c)\tilde{\mathbf{B}}^\top\right)^j\right) c\mathbf{q}_s \end{aligned} \tag{4.4}$$

The spectral radius $\rho((1-c)\tilde{\mathbf{B}}^\top) = (1-c) < 1$ when $0 < c < 1$ since $\tilde{\mathbf{B}}^\top$ is a column stochastic matrix and its largest eigenvalue is 1 [115]. Therefore, $\lim_{k\to\infty}((1-c)\tilde{\mathbf{B}}^\top)^k \mathbf{h}^{(0)} = \mathbf{0}$ and $\lim_{k\to\infty} \mathbf{h}^{(k)}$ converges as follows:

$$\lim_{k\to\infty} \mathbf{h}^{(k)} = \mathbf{0} + \lim_{k\to\infty} \left(\sum_{j=0}^{k-1} \left((1-c)\tilde{\mathbf{B}}^\top\right)^j\right) c\mathbf{q}_s = c\left(\mathbf{I} - (1-c)\tilde{\mathbf{B}}^\top\right)^{-1} \mathbf{q}_s.$$

In the above equation, $\sum_{j=0}^{\infty}((1-c)\tilde{\mathbf{B}}^\top)^j$ is a geometric series of the matrix $(1-c)\tilde{\mathbf{B}}^\top$, and the series converges to $(\mathbf{I} - (1-c)\tilde{\mathbf{B}}^\top)^{-1}$ since the spectral radius of $(1-c)\tilde{\mathbf{B}}^\top$ is less than one. Note that the inverse matrix is a non-negative matrix whose entries are positive or zero because the matrix is the sum of non-negative matrices (i.e., $\sum_{j=0}^{\infty}((1-c)\tilde{\mathbf{B}}^\top)^j$). Hence, each entry of $\mathbf{h}$ is non-negative (i.e., $\mathbf{h}_u \geq 0$ for any node u). □

Error analysis of SRWR-Iter. We show that the error δ of SRWR-Iter monotonically decrease over iterations using the following lemma.

Lemma 4.2 (Error analysis of SRWR-Iter)**.** *Suppose* $\mathbf{h} = [\mathbf{r}^+; \mathbf{r}^-]^\top$*, and* $\mathbf{h}^{(k)}$ *is the*

result of k-th iteration in SRWR-ITER. Let $\delta^{(k)}$ *denote the error* $\|\mathbf{h}^{(k)} - \mathbf{h}^{(k-1)}\|_1$. *Then* $\delta^{(k)} \leq 2(1-c)^k$, *and the estimated number* T *of iterations for convergence is* $\log_{1-c} \frac{\varepsilon}{2}$ *where* ε *is an error tolerance, and* c *is the restart probability.*

Proof. According to Equation (4.4), $\delta^{(k)}$ is represented as follows:

$$\begin{aligned}\delta^{(k)} = \|\mathbf{h}^{(k)} - \mathbf{h}^{(k-1)}\|_1 &= (1-c)\|\tilde{\mathbf{B}}^\top(\mathbf{h}^{(k-1)} - \mathbf{h}^{(k-2)})\|_1 \\ &\leq (1-c)\|\tilde{\mathbf{B}}^\top\|_1\|\mathbf{h}^{(k-1)} - \mathbf{h}^{(k-2)}\|_1 \\ &= (1-c)\|\mathbf{h}^{(k-1)} - \mathbf{h}^{(k-2)}\|_1 = (1-c)\delta^{(k-1)}\end{aligned}$$

Note that $\|\tilde{\mathbf{B}}^\top\|_1 = 1$ since $\tilde{\mathbf{B}}^\top$ is column stochastic as described in Theorem 4.1. Hence, $\delta^{(k)} \leq (1-c)\delta^{(k-2)} \leq \cdots \leq (1-c)^k\delta^{(1)}$. Since $\delta^{(1)} = \|\mathbf{h}^{(1)} - \mathbf{h}^{(0)}\|_1 \leq \|\mathbf{h}^{(1)}\|_1 + \|\mathbf{h}^{(0)}\|_1 = 2$, $\delta^{(k)} \leq 2(1-c)^k$. Note that when $\delta^{(k)} \leq \varepsilon$, the iteration of SRWR-ITER is terminated. Thus, for $k \leq \log_{1-c} \frac{\varepsilon}{2}$, the iteration is terminated, and the number T of iterations for convergence is estimated at $\log_{1-c} \frac{\varepsilon}{2}$. □

Properties of SRWR. We discuss the properties of our ranking model SRWR to answer the following questions: 1) Is the resulting SRWR score vector a probability distribution (Property 1)? 2) Is the signed random surfer able to visit all nodes in a network which is strongly connected (Property 2)? and 3) Does SRWR work on unsigned networks as well (Property 3)?

Property 1. *Consider the recursive equation* $\mathbf{p} = (1-c)|\tilde{\mathbf{A}}|^\top\mathbf{p} + c\mathbf{q}$ *where* $\mathbf{p} = \mathbf{r}^+ + \mathbf{r}^-$ *and* $|\tilde{\mathbf{A}}|^\top$ *is a column stochastic matrix. Then* $\mathbf{1}^\top\mathbf{p} = \sum_i \mathbf{p}_i = 1$.

Proof. By multiplying both sides by $\mathbf{1}^\top$, the equation is represented as follows:

$$\mathbf{p} = (1-c)|\tilde{\mathbf{A}}|^\top\mathbf{p} + c\mathbf{q} \Leftrightarrow \mathbf{1}^\top\mathbf{p} = (1-c)\mathbf{1}^\top|\tilde{\mathbf{A}}|^\top\mathbf{p} + c\mathbf{1}^\top\mathbf{q}$$

Note that $\mathbf{1}^\top|\tilde{\mathbf{A}}|^\top = (|\tilde{\mathbf{A}}|\mathbf{1})^\top$, and $|\tilde{\mathbf{A}}|$ is a row stochastic matrix; thus, $(|\tilde{\mathbf{A}}|\mathbf{1})^\top = \mathbf{1}^\top$. Hence, the above equation is represented as follows:

$$\mathbf{1}^\top\mathbf{p} = (1-c)\mathbf{1}^\top|\tilde{\mathbf{A}}|^\top\mathbf{p} + c\mathbf{1}^\top\mathbf{q} \Leftrightarrow \mathbf{1}^\top\mathbf{p} = (1-c)\mathbf{1}^\top\mathbf{p} + c \Leftrightarrow \mathbf{1}^\top\mathbf{p} = 1$$

This indicates that the resulting SRWR scores follow a probability distribution. □

Property 2. *Suppose a signed network is strongly connected. Then, all entries of* $\mathbf{r}^+ + \mathbf{r}^-$ *are positive (i.e.,* $\mathbf{r}^+ + \mathbf{r}^- > 0$*).*

Proof. Let $\mathbf{r}^+ + \mathbf{r}^-$ be $\mathbf{p}$. By summing the recursive equations on $\mathbf{r}^+$ and $\mathbf{r}^-$ in Equation (4.3), $\mathbf{p}$ is represented as follows:

$$\mathbf{p} = (1-c)\left(\tilde{\mathbf{A}}_+^\top\mathbf{p} + \tilde{\mathbf{A}}_-^\top\mathbf{p}\right) + c\mathbf{q} \Leftrightarrow \mathbf{p} = (1-c)|\tilde{\mathbf{A}}|^\top\mathbf{p} + c\mathbf{q} \Leftrightarrow \mathbf{p} = \mathbf{G}\mathbf{p}$$

where $|\tilde{\mathbf{A}}| = \tilde{\mathbf{A}}_+ + \tilde{\mathbf{A}}_-$ by Definition 4.3, $\mathbf{G} = (1-c)|\tilde{\mathbf{A}}|^\top + c\mathbf{q}\mathbf{1}^\top$, and $\mathbf{1}^\top\mathbf{p} = \sum_i \mathbf{p}_i = 1$ by Property 1. Note that the graph represented by $\mathbf{G}$ is also strongly connected since the graph of $|\tilde{\mathbf{A}}|$ has the same topology with the original graph which is strongly connected. Moreover, the graph represented by $\mathbf{G}$ has a self-loop at the seed node s due to $c\mathbf{q}\mathbf{1}^\top$. Thus, $\mathbf{G}$ is irreducible and aperiodic. Hence, all entries of $\mathbf{p} = \mathbf{r}^+ + \mathbf{r}^-$ are positive according to Perron-Frobenius theorem [83]. □

Note that $\mathbf{r}_u^+$ (or $\mathbf{r}_u^-$) indicates that the stationary probability of the positive (or negative) surfer visits node u after performing SRWR starting from a seed node. According to Property 2, $\mathbf{r}_u^+ + \mathbf{r}_u^-$ for an arbitrary node u is always positive if a given signed network is strongly connected. That is, the signed random surfer is able to visit node u with probability $\mathbf{r}_u^+ + \mathbf{r}_u^-$ which is always greater than zero.

Property 3. *The result of SRWR on networks containing only positive edges is the same as that of RWR.*

Proof. $\tilde{\mathbf{A}}_+ = \tilde{\mathbf{A}}$ and $\tilde{\mathbf{A}}_- = \mathbf{0}_{n\times n}$ because the adjacency matrix $\mathbf{A}$ only contains positive edges. Also, $\mathbf{r}^- = \mathbf{0}_{n\times 1}$ at the beginning time of Algorithm 6. Equation (4.3) is represented as follows:

$$\mathbf{r}^+ = (1-c)\left(\tilde{\mathbf{A}}^\top \mathbf{r}^+ + \beta \mathbf{0}_{n\times n} \times \mathbf{0}_{n\times 1} + (1-\gamma)\tilde{\mathbf{A}}^\top \mathbf{0}_{n\times 1}\right) + c\mathbf{q}$$
$$\mathbf{r}^- = (1-c)\left(\mathbf{0}_{n\times n} \times \mathbf{r}^+ + \gamma \tilde{\mathbf{A}}^\top \mathbf{0}_{n\times 1} + (1-\beta)\mathbf{0}_{n\times n} \times \mathbf{0}_{n\times 1}\right)$$

Therefore, $\mathbf{r}^- = \mathbf{0}_{n\times 1}$ and $\mathbf{r}^+ = (1-c)\tilde{\mathbf{A}}^\top \mathbf{r}^+ + c\mathbf{q}$. The equation of $\mathbf{r}^+$ is exactly the same as that of RWR. □

This implies that our model SRWR is a generalized version of RWR working on both unsigned and signed networks in the following property.

4.3.3 SRWR-Pre: Preprocessing Algorithm for Signed Random Walk with Restart

We propose SRWR-Pre, a preprocessing algorithm to quickly compute SRWR scores. The iterative approach SRWR-Iter in Algorithm 6 requires multiple matrix-vector multiplications to compute SRWR scores whenever seed node s changes; thus the iterative method is not fast enough when we require SRWR scores for any pair of nodes in large-scale signed networks. Our goal is to directly compute SRWR scores from precomputed intermediate data without iterations. We exploit the following ideas for our preprocessing method:

- The positive and negative SRWR score vectors $\mathbf{r}^+$ and $\mathbf{r}^-$ are obtained by solving linear systems (Section 4.3.3.1).

- The adjacency matrix of real-world graphs is permuted so that it contains a large but easy-to-invert block diagonal matrix as shown in Figure 4.6 (Section 4.3.3.2).
- The block elimination approach efficiently solves a linear system on a matrix if it has an easy-to-invert sub-matrix (Section 4.3.3.3).

Our preprocessing method comprises two phases: preprocessing phase (Algorithm 7) and query phase (Algorithm 8). The preprocessing phase preprocesses a given signed adjacency matrix into several sub-matrices required in the query phase to compute SRWR scores w.r.t. seed node s. Note that the preprocessing phase is performed once, and the query phase is run for each seed node. The starting vector $\mathbf{q}$ in Equation (4.3) is called an SRWR query, and $\mathbf{r}^+$ and $\mathbf{r}^-$ are the results corresponding to the query $\mathbf{q}$. The query vector $\mathbf{q}$ is determined by the seed node s, and $\mathbf{r}^+$ and $\mathbf{r}^-$ are distinct for each SRWR query. To exploit sparsity of graphs, we save all matrices in a sparse matrix format such as compressed column storage [100] which stores only non-zero entries and their locations.

4.3.3.1 Formulation of Signed Random Walk with Restart as Linear Systems

We first represent linear systems related to $\mathbf{r}^+$ and $\mathbf{r}^-$. Let $\mathbf{p}$ be the sum of $\mathbf{r}^+$ and $\mathbf{r}^-$ (i.e., $\mathbf{p} = \mathbf{r}^+ + \mathbf{r}^-$). Then, $\mathbf{p}$ is the solution of the following linear system:

$$|\mathbf{H}|\mathbf{p} = c\mathbf{q} \Leftrightarrow \mathbf{p} = c|\mathbf{H}|^{-1}\mathbf{q} \tag{4.5}$$

where $|\mathbf{H}| = \mathbf{I} - (1-c)|\tilde{\mathbf{A}}|^\top$ and $|\tilde{\mathbf{A}}| = \tilde{\mathbf{A}}_+ + \tilde{\mathbf{A}}_-$. The proof of Equation (4.5) is presented in Lemma 4.3. The linear system for $\mathbf{r}^-$ is given by the following equation:

$$\mathbf{T}\mathbf{r}^- = (1-c)\tilde{\mathbf{A}}_-^\top \mathbf{p} \Leftrightarrow \mathbf{r}^- = (1-c)\left(\mathbf{T}^{-1}(\tilde{\mathbf{A}}_-^\top \mathbf{p})\right) \tag{4.6}$$

where $\mathbf{T} = \mathbf{I} - (1-c)(\gamma\tilde{\mathbf{A}}_+^\top - \beta\tilde{\mathbf{A}}_-^\top)$, and γ and β are balance attenuation factors. Theorem 4.2 shows the proof of Equation (4.6). Based on the aforementioned linear systems in Equations (4.5) and (4.6), $\mathbf{r}^-$ and $\mathbf{r}^+$ for a given seed node s are computed as follows:

1. Set a query vector $\mathbf{q}$ whose s-th element is 1 and all other elements are 0.
2. Solve the linear system in Equation (4.5) to obtain the solution $\mathbf{p}$.
3. Compute $\mathbf{r}^-$ by solving the linear system in Equation (4.6).
4. Compute $\mathbf{r}^+ = \mathbf{p} - \mathbf{r}^-$.

Lemma 4.3. *Suppose that* $\mathbf{p} = \mathbf{r}^+ + \mathbf{r}^-$, $|\mathbf{H}| = \mathbf{I} - (1-c)|\tilde{\mathbf{A}}|^\top$ *and* $|\tilde{\mathbf{A}}| = \tilde{\mathbf{A}}_+ + \tilde{\mathbf{A}}_-$. *Then,* $\mathbf{p}$ *is the solution of the following linear system:*

$$|\mathbf{H}|\mathbf{p} = c\mathbf{q} \Leftrightarrow \mathbf{p} = c|\mathbf{H}|^{-1}\mathbf{q}$$

Proof. According to the result in Property 2, the recursive equation for $\mathbf{p}$ is represented as follows:

$$\mathbf{p} = (1-c)|\tilde{\mathbf{A}}|^\top \mathbf{p} + c\mathbf{q}$$

where $|\tilde{\mathbf{A}}| = \tilde{\mathbf{A}}_+ + \tilde{\mathbf{A}}_-$ is the row-normalized matrix of $|\mathbf{A}|$. The linear system for $\mathbf{p}$ is

represented by moving $(1-c)|\tilde{\mathbf{A}}|^\top \mathbf{p}$ to the left side as follows:

$$\left(\mathbf{I} - (1-c)|\tilde{\mathbf{A}}|^\top\right)\mathbf{p} = c\mathbf{q} \Leftrightarrow |\mathbf{H}|\mathbf{p} = c\mathbf{q}$$

where $|\mathbf{H}|$ is $\mathbf{I} - (1-c)|\tilde{\mathbf{A}}|^\top$. Note that $|\mathbf{H}|$ is invertible when $0 < c < 1$ because it is strictly diagonally dominant [116]. Hence, $\mathbf{p} = c|\mathbf{H}|^{-1}\mathbf{q}$. □

Theorem 4.2. *The SRWR score vectors* $\mathbf{r}^+$ *and* $\mathbf{r}^-$ *from Equation* (4.3) *are represented as follows:*

$$\mathbf{r}^+ = \mathbf{p} - \mathbf{r}^-$$
$$\mathbf{r}^- = (1-c)\left(\mathbf{T}^{-1}(\tilde{\mathbf{A}}_-^\top \mathbf{p})\right)$$

where $\mathbf{p} = c|\mathbf{H}|^{-1}\mathbf{q}$, $\mathbf{T} = \mathbf{I} - (1-c)(\gamma\tilde{\mathbf{A}}_+^\top - \beta\tilde{\mathbf{A}}_-^\top)$, *and* γ *and* β *are balance attenuation factors which are between* 0 *and* 1 *(i.e.,* $0 < \gamma, \beta < 1$*).*

Proof. Note that $\mathbf{r}^- = (1-c)(\tilde{\mathbf{A}}_-^\top \mathbf{r}^+ + \gamma\tilde{\mathbf{A}}_+^\top \mathbf{r}^- + (1-\beta)\tilde{\mathbf{A}}_-^\top \mathbf{r}^-)$ by Equation (4.3), and $\mathbf{r}^+ = \mathbf{p} - \mathbf{r}^-$ according to Lemma 4.3. The equation for $\mathbf{r}^-$ is represented by plugging $\mathbf{r}^+ = \mathbf{p} - \mathbf{r}^-$ as follows:

$$\mathbf{r}^- = (1-c)\left(\tilde{\mathbf{A}}_-^\top \mathbf{p} - \tilde{\mathbf{A}}_-^\top \mathbf{r}^- + \gamma\tilde{\mathbf{A}}_+^\top \mathbf{r}^- + (1-\beta)\tilde{\mathbf{A}}_-^\top \mathbf{r}^-\right) \Leftrightarrow$$
$$\mathbf{r}^- = (1-c)\left(\gamma\tilde{\mathbf{A}}_+^\top - \beta\tilde{\mathbf{A}}_-^\top\right)\mathbf{r}^- + (1-c)\tilde{\mathbf{A}}_-^\top \mathbf{p}$$

We move $(1-c)(\gamma\tilde{\mathbf{A}}_+^\top - \beta\tilde{\mathbf{A}}_-^\top)\mathbf{r}^-$ to the left side; then, the above equation is represented as follows:

$$\left(\mathbf{I} - (1-c)(\gamma\tilde{\mathbf{A}}_+^\top - \beta\tilde{\mathbf{A}}_-^\top)\right)\mathbf{r}^- = (1-c)\tilde{\mathbf{A}}_-^\top \mathbf{p} \Leftrightarrow \mathbf{T}\mathbf{r}^- = (1-c)\tilde{\mathbf{A}}_-^\top \mathbf{p}$$

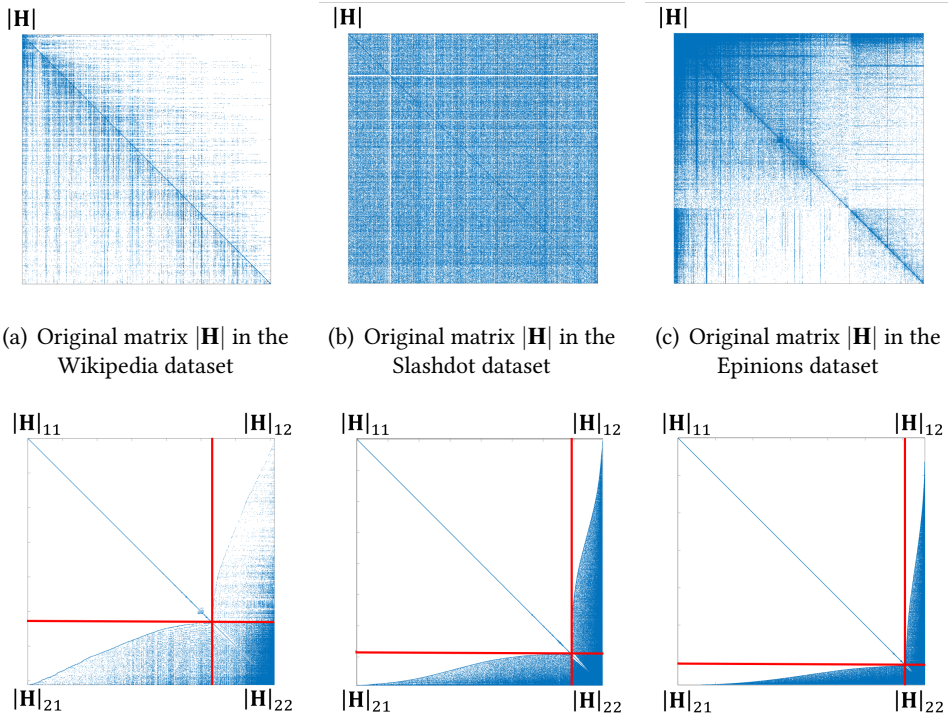
where $\mathbf{T}$ is $\mathbf{I} - (1-c)(\gamma\tilde{\mathbf{A}}_+^\top - \beta\tilde{\mathbf{A}}_-^\top)$. Note that the matrix $\mathbf{T}$ is strictly diagonally dominant when $0 < c < 1$ and $0 < \gamma, \beta < 1$; thus, $\mathbf{T}$ is invertible. Hence, $\mathbf{r}^- = (1-c)(\mathbf{T}^{-1}(\tilde{\mathbf{A}}_-^\top \mathbf{p}))$. $\mathbf{r}^+$ is obtained by computing $\mathbf{r}^+ = \mathbf{p} - \mathbf{r}^-$. □

One naive approach (Inversion) for SRWR score vectors $\mathbf{r}^+$ and $\mathbf{r}^-$ based on the linear systems in Equations (4.5) and (4.6) is to precompute the inverse of the matrices $|\mathbf{H}|$ and $\mathbf{T}$. However, this approach is impractical for large-scale graphs since inverting a matrix requires $O(n^3)$ time and $O(n^2)$ space where n is the dimensions of the matrix. Another approach (LU) is to obtain the inverse of LU factors of $|\mathbf{H}|$ and $\mathbf{T}$ after reordering the matrices in the order of node degrees as suggested in [60] (i.e., $\mathbf{p} = c(\mathbf{U}_\mathbf{p}^{-1}(\mathbf{L}_\mathbf{p}^{-1}\mathbf{q}))$; $\mathbf{r}^- = (1-c)(\mathbf{U}_{\mathbf{r}^-}^{-1}(\mathbf{L}_{\mathbf{r}^-}^{-1}(\tilde{\mathbf{A}}_-^\top \mathbf{p})))$ where $|\mathbf{H}|^{-1} = \mathbf{U}_\mathbf{p}^{-1}\mathbf{L}_\mathbf{p}^{-1}$ and $\mathbf{T}^{-1} = \mathbf{U}_{\mathbf{r}^-}^{-1}\mathbf{L}_{\mathbf{r}^-}^{-1}$). Although LU is more efficient than Inversion in terms of time and space as shown in Figure 4.13, LU still has a performance issue due to $O(n^3)$ time and $O(n^2)$ space complexities. On the other hand, our preprocessing method SRWR-Pre is faster and more memory efficient than Inversion and LU as we will see in Section 4.4.7.

4.3.3.2 Node Reordering based on Hub-and-Spoke Structure

SRWR-Pre permutes the matrices $|\mathbf{H}|$ and $\mathbf{T}$ using a reordering technique based on hub-and-spoke structure. Previous works [54, 80] have exploited the reordering technique to reduce computational cost of graph operations in real-world graphs. We also adopt the node reordering based on hub-and-spoke structure to efficiently solve the linear systems in Equations (4.5) and (4.6).

The hub-and-spoke structure indicates that most real-world graphs follow power-law degree distribution with few hubs (very high degree nodes) and majority of spokes (low degree nodes). The structure has been utilized to concentrate entries



(a) Original matrix $|\mathbf{H}|$ in the Wikipedia dataset

(b) Original matrix $|\mathbf{H}|$ in the Slashdot dataset

(c) Original matrix $|\mathbf{H}|$ in the Epinions dataset

(d) Reordered matrix $|\mathbf{H}|$ in the Wikipedia dataset

(e) Reordered matrix $|\mathbf{H}|$ in the Slashdot dataset

(f) Reordered matrix $|\mathbf{H}|$ in the Epinions dataset

Figure 4.6: **Result of node reordering on each signed network.** (a), (b), and (c) are the original matrix $|\mathbf{H}|$ before node reordering in the Wikipedia, the Slashdot, and the Epinions datasets, respectively. (d), (e) and (f) present $|\mathbf{H}|$ reordered by the hub-and-spoke method. Note that $\mathbf{T}$ is also reordered equivalently to $|\mathbf{H}|$ since they have the same sparsity pattern. $|\mathbf{H}|_{11}$ and $\mathbf{T}_{11}$ are block diagonal.

of an adjacency matrix by reordering nodes as shown in Figure 4.6. Any reordering method based on the hub-and-spoke structure can be utilized for the purpose; in this paper, we use SlashBurn [98, 117] as a hub-and-spoke reordering method because it shows the best performance in concentrating entries of an adjacency matrix (see the details in Appendix A.1).

We reorder nodes of the signed adjacency matrix $\mathbf{A}$ so that reordered matrix contains a large but easy-to-invert submatrix such as block diagonal matrix as shown in Figure 4.6. We then compute $|\mathbf{H}| = \mathbf{I} - (1-c)(\tilde{\mathbf{A}}_+^\top + \tilde{\mathbf{A}}_-^\top)$ and $\mathbf{T} = \mathbf{I} - (1-c)(\gamma\tilde{\mathbf{A}}_+^\top - \beta\tilde{\mathbf{A}}_-^\top)$. Note that $|\mathbf{H}|$ and $\mathbf{T}$ have the same sparsity pattern as the reordered adjacency matrix $\mathbf{A}^\top$ except for the diagonal part. Hence, $|\mathbf{H}|$ and $\mathbf{T}$ are partitioned as follows:

$$|\mathbf{H}| = \begin{bmatrix} |\mathbf{H}|_{11} & |\mathbf{H}|_{12} \\ |\mathbf{H}|_{21} & |\mathbf{H}|_{22} \end{bmatrix}, \mathbf{T} = \begin{bmatrix} \mathbf{T}_{11} & \mathbf{T}_{12} \\ \mathbf{T}_{21} & \mathbf{T}_{22} \end{bmatrix}. \tag{4.7}$$

Let n_1 and n_2 denote the number of spokes and hubs, respectively (see the details in Appendix A.1). Then $|\mathbf{H}|_{11}$ and $\mathbf{T}_{11}$ are $n_1 \times n_1$ matrices, $|\mathbf{H}|_{12}$ and $\mathbf{T}_{12}$ are $n_1 \times n_2$ matrices, $|\mathbf{H}|_{21}$ and $\mathbf{T}_{21}$ are $n_2 \times n_1$ matrices, and $|\mathbf{H}|_{22}$ and $\mathbf{T}_{22}$ are $n_2 \times n_2$ matrices. The linear systems for $|\mathbf{H}|$ and $\mathbf{T}$ in Equations (4.5) and (4.6) are represented as follows:

$$|\mathbf{H}|\mathbf{p} = c\mathbf{q} \Leftrightarrow \begin{bmatrix} |\mathbf{H}|_{11} & |\mathbf{H}|_{12} \\ |\mathbf{H}|_{21} & |\mathbf{H}|_{22} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} = c \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \end{bmatrix} \tag{4.8}$$

$$\mathbf{T}\mathbf{r}^- = (1-c)\mathbf{t} \Leftrightarrow \begin{bmatrix} \mathbf{T}_{11} & \mathbf{T}_{12} \\ \mathbf{T}_{21} & \mathbf{T}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{r}_1^- \\ \mathbf{r}_2^- \end{bmatrix} = (1-c) \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \end{bmatrix} \tag{4.9}$$

where $\mathbf{t} = \tilde{\mathbf{A}}_-^\top \mathbf{p}$ is an $n \times 1$ vector.

4.3.3.3 Block Elimination for Solving Linear Systems

The solutions of the partitioned linear systems in Equations (4.8) and (4.9) are obtained by the following equations:

$$\mathbf{p} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} = \begin{bmatrix} |\mathbf{H}|_{11}^{-1}(c\mathbf{q}_1 - |\mathbf{H}|_{12}\mathbf{p}_2) \\ c(\mathbf{S}_{|\mathbf{H}|}^{-1}(\mathbf{q}_2 - |\mathbf{H}|_{21}(|\mathbf{H}|_{11}^{-1}(\mathbf{q}_1)))) \end{bmatrix} \tag{4.10}$$

$$\mathbf{r}^- = \begin{bmatrix} \mathbf{r}_1^- \\ \mathbf{r}_2^- \end{bmatrix} = \begin{bmatrix} \mathbf{T}_{11}^{-1}((1-c)\mathbf{t}_1 - \mathbf{T}_{12}\mathbf{r}_2^-) \\ (1-c)(\mathbf{S}_{\mathbf{T}}^{-1}(\mathbf{t}_2 - \mathbf{T}_{21}(\mathbf{T}_{11}^{-1}(\mathbf{t}_1)))) \end{bmatrix} \tag{4.11}$$

where $\mathbf{S}_{|\mathbf{H}|} = |\mathbf{H}|_{22} - |\mathbf{H}|_{21}|\mathbf{H}|_{11}^{-1}|\mathbf{H}|_{12}$ is the Schur complement of $|\mathbf{H}|_{11}$ and $\mathbf{S}_{\mathbf{T}} = \mathbf{T}_{22} - \mathbf{T}_{21}\mathbf{T}_{11}^{-1}\mathbf{T}_{12}$ is the Schur complement of $\mathbf{T}_{11}$. Equations (4.10) and (4.11) are derived by applying block elimination described in Lemma 4.4 to the partitioned linear systems in Equations (4.8) and (4.9), respectively. Note that the sub-matrices $|\mathbf{H}|_{11}$ and $\mathbf{T}_{11}$ are invertible when $0 < c < 1$ and $0 < \gamma, \beta < 1$ since they are strictly diagonally dominant. If all matrices in Equations (4.10) and (4.11) are precomputed, then the SRWR score vectors $\mathbf{r}^+$ and $\mathbf{r}^-$ are efficiently and directly computed from the precomputed matrices.

Lemma 4.4 (Block Elimination [99]). *Suppose a linear system* $\mathbf{Ax} = \mathbf{b}$ *is partitioned as follows:*

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$$

where $\mathbf{A}_{11}$ *and* $\mathbf{A}_{22}$ *are square matrices. If the sub-matrix* $\mathbf{A}_{11}$ *is invertible, then the*

Algorithm 7: Preprocessing phase of SRWR-Pre

Input: signed adjacency matrix: $\mathbf{A}$, restart probability: c, balance attenuation factors: β and γ
Output: preprocessed matrices from $|\mathbf{H}|$ and $\mathbf{T}$, negative semi-row normalized matrix $\tilde{\mathbf{A}}_-$

1: reorder $\mathbf{A}$ using the hub-and-spoke reordering method [98, 117]
2: compute $\tilde{\mathbf{A}}_+$ and $\tilde{\mathbf{A}}_-$ from $\mathbf{A}$ using Algorithm 5
3: compute $|\mathbf{H}|$ and $\mathbf{T}$, i.e., $|\mathbf{H}| = \mathbf{I} - (1-c)|\tilde{\mathbf{A}}|^\top$ and $\mathbf{T} = \mathbf{I} - (1-c)(\gamma\tilde{\mathbf{A}}_+^\top - \beta\tilde{\mathbf{A}}_-^\top)$
4: partition $|\mathbf{H}|$ into $|\mathbf{H}|_{11}$, $|\mathbf{H}|_{12}$, $|\mathbf{H}|_{21}$, $|\mathbf{H}|_{22}$, and compute $|\mathbf{H}|_{11}^{-1}$
5: partition $\mathbf{T}$ into $\mathbf{T}_{11}$, $\mathbf{T}_{12}$, $\mathbf{T}_{21}$, $\mathbf{T}_{22}$, and compute $\mathbf{T}_{11}^{-1}$
6: compute the Schur complement of $|\mathbf{H}|_{11}$, i.e., $\mathbf{S}_{|\mathbf{H}|} = |\mathbf{H}|_{22} - |\mathbf{H}|_{21}|\mathbf{H}|_{11}^{-1}|\mathbf{H}|_{12}$
7: compute the Schur complement of $\mathbf{T}_{11}$, i.e., $\mathbf{S}_{\mathbf{T}} = \mathbf{T}_{22} - \mathbf{T}_{21}\mathbf{T}_{11}^{-1}\mathbf{T}_{12}$
8: compute the inverse of LU factors of $\mathbf{S}_{|\mathbf{H}|}$, i.e., $\mathbf{S}_{|\mathbf{H}|}^{-1} = \mathbf{U}_{|\mathbf{H}|}^{-1}\mathbf{L}_{|\mathbf{H}|}^{-1}$
9: compute the inverse of LU factors of $\mathbf{S}_{\mathbf{T}}$, i.e., $\mathbf{S}_{\mathbf{T}}^{-1} = \mathbf{U}_{\mathbf{T}}^{-1}\mathbf{L}_{\mathbf{T}}^{-1}$
10: **return** preprocessed matrices from $|\mathbf{H}|$: $\mathbf{L}_{|\mathbf{H}|}^{-1}$, $\mathbf{U}_{|\mathbf{H}|}^{-1}$, $|\mathbf{H}|_{11}^{-1}$, $|\mathbf{H}|_{12}$, and $|\mathbf{H}|_{21}$
preprocessed matrices from $\mathbf{T}$: $\mathbf{L}_{\mathbf{T}}^{-1}$, $\mathbf{U}_{\mathbf{T}}^{-1}$, $\mathbf{T}_{11}^{-1}$, $\mathbf{T}_{12}$, and $\mathbf{T}_{21}$
negative semi-row normalized matrix $\tilde{\mathbf{A}}_-$

solution $\mathbf{x}$ *is represented as follows:*

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11}^{-1}(\mathbf{b}_1 - \mathbf{A}_{12}\mathbf{x}_2) \\ \mathbf{S}^{-1}(\mathbf{b}_2 - \mathbf{A}_{21}(\mathbf{A}_{11}^{-1}(\mathbf{b}_1))) \end{bmatrix}$$

where $\mathbf{S} = \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12}$ *is the Schur complement of* $\mathbf{A}_{11}$. ■

Lemma 4.4 implies that a partitioned linear system is efficiently solved if it contains an easy-to-invert sub-matrix and the dimension of the Schur complement is small. Note that inverting $\mathbf{H}_{11}$ and $\mathbf{T}_{11}$ is trivial because they are block diagonal matrices as shown in Figure 4.6. Also, the dimension of $\mathbf{S}_{|\mathbf{H}|}$ and $\mathbf{S}_{\mathbf{T}}$ is n_2 where n_2 is the number of hubs and most real-world graphs have a small number of hubs compared to the number of nodes (see Table 4.3).

Preprocessing phase (Algorithm 7). Our preprocessing phase precomputes the matrices exploited for computing SRWR scores in the query phase. Our algorithm first reorders nodes of a given signed adjacency matrix $\mathbf{A}$ using the hub-and-spoke

Algorithm 8: Query phase of SRWR-Pre

Input: seed node: s, preprocessed matrices from Algorithm 7
Output: trustworthiness SRWR score vector: $\mathbf{r}$

1: create $\mathbf{q}$ whose s-th entry is 1 and the others are 0, and partition $\mathbf{q}$ into $\mathbf{q}_1$ and $\mathbf{q}_2$
2: compute $\mathbf{p}_2 = c(\mathbf{U}_{|\mathbf{H}|}^{-1}(\mathbf{L}_{|\mathbf{H}|}^{-1}(\mathbf{q}_2 - |\mathbf{H}|_{21}(|\mathbf{H}|_{11}^{-1}\mathbf{q}_1))))$
3: compute $\mathbf{p}_1 = |\mathbf{H}|_{11}^{-1}(c\mathbf{q}_1 - |\mathbf{H}|_{12}\mathbf{p}_2)$
4: create $\mathbf{p}$ by concatenating $\mathbf{p}_1$ and $\mathbf{p}_2$
5: compute $\mathbf{t} = \tilde{\mathbf{A}}_-^\top \mathbf{p}$, and partition it into $\mathbf{t}_1$ and $\mathbf{t}_2$
6: compute $\mathbf{r}_2^- = (1-c)(\mathbf{U}_\mathbf{T}^{-1}(\mathbf{L}_\mathbf{T}^{-1}(\mathbf{t}_2 - \mathbf{T}_{21}(\mathbf{T}_{11}^{-1}\mathbf{t}_1))))$
7: compute $\mathbf{r}_1^- = \mathbf{T}_{11}^{-1}((1-c)\mathbf{t}_1 - \mathbf{T}_{12}\mathbf{r}_2^-))$
8: create $\mathbf{r}^-$ by concatenating $\mathbf{r}_1^-$ and $\mathbf{r}_2^-$
9: compute $\mathbf{r}^+ = \mathbf{p} - \mathbf{r}^-$
10: compute $\mathbf{r} = \mathbf{r}^+ - \mathbf{r}^-$
11: **return** $\mathbf{r}$

reordering method, and performs semi-normalization on $\mathbf{A}$ to obtain $\tilde{\mathbf{A}}_+$ and $\tilde{\mathbf{A}}_-$ using Algorithm 5 (lines 1∼2). Then our algorithm computes $|\mathbf{H}|$ and $\mathbf{T}$, and partitions the matrices as shown in Figure 4.6 (lines 3∼5). Our algorithm calculates the inverses of $|\mathbf{H}|_{11}$ and $\mathbf{T}_{11}$, and computes the Schur complements of $|\mathbf{H}|_{11}$ and $\mathbf{T}_{11}$ (lines 4∼7). When we compute $\mathbf{S}_{|\mathbf{H}|}^{-1}$ and $\mathbf{S}_\mathbf{T}^{-1}$, we invert the LU factors of $\mathbf{S}_{|\mathbf{H}|}$ and $\mathbf{S}_\mathbf{T}$ (lines 8 and 9) because this approach is faster and more memory efficient than directly inverting $\mathbf{S}_{|\mathbf{H}|}$ and $\mathbf{S}_\mathbf{T}$ as suggested in [54].

Query phase (Algorithm 8). Our query phase computes SRWR score vectors $\mathbf{r}^+$ and $\mathbf{r}^-$ for a given seed node s using precomputed matrices from Algorithm 7. Our algorithm first creates a starting vector $\mathbf{q}$ whose entry at the index of the seed node s is 1 and otherwise 0, and partitions $\mathbf{q}$ into $\mathbf{q}_1$ and $\mathbf{q}_2$ (line 1). We then compute $\mathbf{p}_2$ and $\mathbf{p}_1$ based on Equation (4.10), and concatenate the vectors to obtain $\mathbf{p}$ (lines 2∼4). Our algorithm calculates $\mathbf{t} = \tilde{\mathbf{A}}_-^\top \mathbf{p}$ and partitions $\mathbf{t}$ into $\mathbf{t}_1$ and $\mathbf{t}_2$ (line 5). We compute $\mathbf{r}_2^-$ and $\mathbf{r}_1^-$ based on Equation (4.11), and concatenate the vectors to obtain $\mathbf{r}^-$ (lines 6∼8). After computing $\mathbf{r}^+ = \mathbf{p} - \mathbf{r}^-$ to obtain $\mathbf{r}^+$ (line 9), we obtain $\mathbf{r} = \mathbf{r}^+ - \mathbf{r}^-$ (line 10).

Table 4.2: **Space complexity of each preprocessed matrix from Algorithm 7.** Note that m is the number of edges of the input graph; n_2 is the number of hubs, and n_{1i} is the number of nodes in i-th block where b blocks in $|\mathbf{H}|_{11}$ (or $\mathbf{T}_{11}$) are identified by the hub-and-spoke reordering method.

Matrix	Space Complexity
$\tilde{\mathbf{A}}_{-}$, $\|\mathbf{H}\|_{12}$, $\|\mathbf{H}\|_{21}$, $\mathbf{T}_{12}$, and $\mathbf{T}_{21}$	$O(m)$
$\|\mathbf{H}\|_{11}^{-1}$, and $\mathbf{T}_{11}^{-1}$	$O(\sum_{i=1}^{b} n_{1i}^2) = O(m)$
$\mathbf{L}_{\|\mathbf{H}\|}^{-1}$, $\mathbf{U}_{\|\mathbf{H}\|}^{-1}$, $\mathbf{L}_{\mathbf{T}}^{-1}$, and $\mathbf{U}_{\mathbf{T}}^{-1}$	$O(n_2^2)$

4.3.3.4 Space and Time Complexities of SRWR-Pre

Lemma 4.5 (Space Complexity of SRWR-Pre). *The space complexity of the preprocessed matrices from SRWR-Pre is $O(n_2^2 + m)$ where n_2 is the number of hubs and m is the number of edges in the graph.*

Proof. The space complexity of each preprocessed matrix is summarized in Table 4.2. $\tilde{\mathbf{A}}_{-}$, $|\mathbf{H}|_{12}$, $|\mathbf{H}|_{21}$, $\mathbf{T}_{12}$, and $\mathbf{T}_{21}$ are sparse matrices, and constructed from the input graph; hence, the space complexity is bounded by the number of edges (i.e., $O(m)$). Note that $|\mathbf{H}|$ and $\mathbf{T}$ have the same sparsity pattern; hence, $|\mathbf{H}|_{11}$ and $\mathbf{T}_{11}$ identified by [98, 117] have the same b blocks. The i-th block in $|\mathbf{H}|_{11}^{-1}$ (or $\mathbf{T}_{11}^{-1}$) contains n_{1i}^2 non-zeros; therefore, $|\mathbf{H}|_{11}^{-1}$ and $\mathbf{T}_{11}^{-1}$ require $O(\sum_{i=1}^{b} n_{1i}^2)$ space, respectively. Since the dimension of $\mathbf{L}_{|\mathbf{H}|}^{-1}$, $\mathbf{U}_{|\mathbf{H}|}^{-1}$, $\mathbf{L}_{\mathbf{T}}^{-1}$, and $\mathbf{U}_{\mathbf{T}}^{-1}$ is n_2, they require $O(n_2^2)$ space. □

Note that the blocks in $|\mathbf{H}|_{11}$ (or $\mathbf{T}_{11}$) are discovered by the reordering method [98, 117] as briefly described in Appendix A.1. In real-world graphs, $\sum_{i=1}^{b} n_{1i}^2$ can be bounded by $O(m)$ as shown in [54]. Hence, we assume that the space complexity of $|\mathbf{H}|_{11}^{-1}$ and $\mathbf{T}_{11}^{-1}$ is $O(m)$ for simplicity.

Lemma 4.6 (Time Complexity of Preprocessing Phase in SRWR-Pre). *The preprocessing phase in Algorithm 7 takes $O(T(m + n\log n) + n_2^3 + mn_2)$ where $T = \lceil \frac{n_2}{kn} \rceil$ is the number of iterations, and k is the hub selection ratio in the hub-and-spoke reordering*

method [98, 117].

Proof. We only consider the main factors of the time complexity of Algorithm 7 in this proof. The hub-and-spoke reordering method takes $O(T(m+n\log n))$ time (line 1) where T is $\lceil \frac{n_2}{kn} \rceil$ which is proved in [98, 117]. Computing the Schur complement of $|\mathbf{H}|_{11}$ takes $O(n_2^2+mn_2)$ because it takes $O(mn_2)$ to compute $\mathbf{P}_1 = |\mathbf{H}|_{11}^{-1}|\mathbf{H}|_{12}$ and $\mathbf{P}_2 = |\mathbf{H}|_{21}\mathbf{P}_1$ by Lemma A.1, and $O(n_2^2)$ to compute $|\mathbf{H}|_{22}-\mathbf{P}_2$ (line 6). It takes $O(n_2^3)$ to compute the inverse of the LU factors (line 8). Note that computing $|\mathbf{H}|_{11}^{-1}$(line 4) requires $O(\sum_{i=1}^{b} n_{1i}^3)$ time where it takes n_{1i}^3 to obtain the inverse of i-th block. In real-world networks, the size n_{1i} of each block is much smaller than the number n_2 of hubs; thus, we assume that $\sum_{i=1}^{b} n_{1i}^3 \ll n_2^3$ [54]. Hence, the time complexity of preprocessing $|\mathbf{H}|$ is $O(T(m+n\log n)+n_2^3+mn_2)$. Note that the time complexity of preprocessing $\mathbf{T}$ is included into that of preprocessing $|\mathbf{H}|$ since $\mathbf{T}$ and $|\mathbf{H}|$ have the same sparsity pattern. □

Lemma 4.7 (Time Complexity of Query Phase in SRWR-Pre). *The query phase in Algorithm 8 takes $O(n_2^2+n+m)$ time.*

Proof. We only consider the main factors of the time complexity of Algorithm 8 in this proof. It takes $O(n_2^2+m)$ to compute $\mathbf{p}_2$ since it takes $O(n_2+m)$ to compute $\tilde{\mathbf{q}}_2 = \mathbf{q}_2 - |\mathbf{H}|_{21}(|\mathbf{H}|_{11}^{-1}\mathbf{q}_1)$, and $O(n_2^2)$ to compute $\mathbf{U}_{|\mathbf{H}|}^{-1}(\mathbf{L}_{|\mathbf{H}|}^{-1}\tilde{\mathbf{q}}_2)$ (line 2). It takes $O(n)$ time to concatenate the partitioned vectors (lines 4 and 8) and compute $\mathbf{r}^+$ and $\mathbf{r}$ (lines 9 and 10). Hence, the total time complexity of the query phase is $O(n_2^2+n+m)$. □

4.4 Experiments

We evaluate the effectiveness of SRWR compared to existing ranking methods. Since there is no ground-truth of personalized rankings for each node in real-world graphs,

we exploit an indirect way by examining the performance of applications such as link prediction, troll identification, and sign prediction tasks. We also investigate the performance of our approaches in terms of time and space. Based on these settings, we aim to answer the following questions from the experiments:

- **Q1. Link prediction (Section 4.4.2).** How effective is our proposed SRWR model for the link prediction task in signed networks?
- **Q2. User preference preservation (Section 4.4.3).** How well does our model SRWR preserve users' known preferences in personalized rankings in signed networks?
- **Q3. Troll detection (Section 4.4.4).** How well do personalized rankings of SRWR capture trolls who are abnormal users compared to those of other models?
- **Q4. Sign prediction (Section 4.4.5).** How helpful are trustworthiness scores of SRWR for predicting missing signs of edges in signed networks?
- **Q5. Effects of balance attenuation factors (Section 4.4.6).** How effective are the balance attenuation factors of SRWR for applications in signed networks?
- **Q6. Efficiency (Section 4.4.7).** How fast and memory efficient is our preprocessing method SRWR-Pre compared to other baselines?

4.4.1 Experimental Settings

Machines. The experiments on the effectiveness of SRWR in Sections 4.4.2, 4.4.4, 4.4.5 and 4.4.6 are conducted on a PC with Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz and 8GB memory. The experiments on the computational performance of SRWR-Pre

Table 4.3: **Statistics of the datasets used in Chapter 4.** n is the number of nodes and m is the total number of edges. m_+ is the number of positive edges, m_- is the number of negative edges, and n_2 is the number of hubs.

Dataset	n	m	m_+	m_-	n_2
Wikipedia[1]	7,118	103,617	81,285	22,332	1,800
Slashdot[2]	79,120	515,561	392,316	123,245	10,160
Epinions[3]	131,828	841,372	717,667	123,705	10,164

[1] http://snap.stanford.edu/data/wiki-Vote.html
[2] http://dai-labor.de/IRML/datasets
[3] http://www.trustlet.org/wiki/Extended_Epinions_dataset

in Section 4.4.7 are performed on a workstation with a single Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz and 256GB memory.

Datasets. The signed networks for our experiments are summarized in Table 4.3. We use all datasets in the link prediction task, the sign prediction task and the experiments for evaluating the computational performance of the proposed methods (Sections 4.4.2, 4.4.5, and 4.4.7). We use the Slashdot dataset in the troll identification task (Section 4.4.4) since there is a troll list only in the dataset.

Methods. To answer **Q1-4**, we compare our proposed model with Random Walk with Restart (RWR) [21], Modified Random Walk with Restart (M-RWR) [65], Modified Personalized SALSA (M-PSALSA) [118], Personalized Signed spectral Rank (PSR) [63], Personalized Negative Rank (PNR) [63], Troll-Trust Model (TR-TR) [66], TRUST [91], LOGIT [90], and GAUC-OPT [89]. Note that RWR is computed on the absolute adjacency matrix of a signed network. For **Q5**, we compare our model SRWR to H-SRWR which is a version of SRWR without the balance attenuation parameters. For **Q6**, we compare our preprocessing method SRWR-Pre to other baseline methods Inversion and LU mentioned in Section 4.3.3.1 including our iterative method SRWR-Iter.

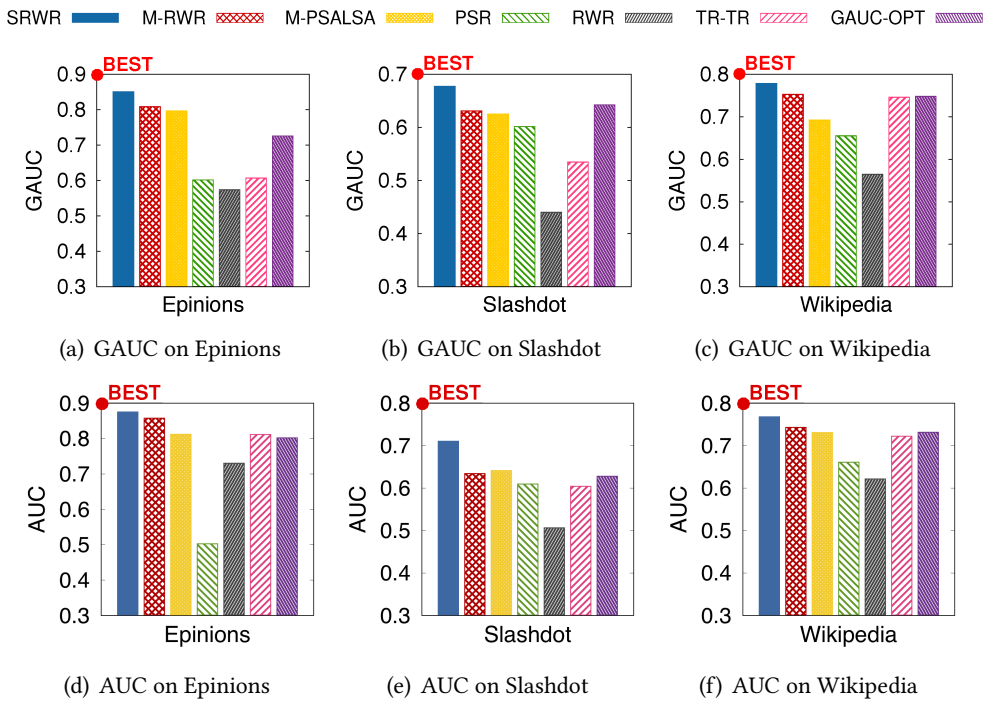
Parameters. There are three hyper-parameters in our ranking model, i.e., restart

probability c and balance attenuation parameters β and γ. We set c to 0.15 for all random walk based approaches including our model for simplicity. To choose β and γ, we perform a grid search over a range $0 \leq \beta, \gamma \leq 1$ by 0.1 (i.e., search (β, γ) in $\mathbf{P} = \{(0.1x, 0.1y) | 0 \leq x, y \leq 10 \text{ and } x, y \in \mathbb{Z}\}$). To select proper parameters, we randomly split a dataset into training, validation, and test sets; and then, we compute personalized rankings based on the training set, and choose the best parameter combination (β, γ) on the validation set with a target metric corresponding to each task. We report results on the test set with the validated parameters. The detailed settings on how to split the dataset and which metric is used are described in each subsection of the corresponding task. The validated parameters of SRWR are summarized as follows:

- Link prediction task (Section 4.4.2): In the Epinions and the Slashdot datasets, $\beta = 0.5$ and $\gamma = 0.8$. In the Wikipedia dataset, $\beta = 0.5$ and $\gamma = 0.5$.
- Troll identification task (Section 4.4.4): In this task, the Slashdot dataset is used as mentioned above, and in the dataset, $\beta = 0.1$ and $\gamma = 1.0$.
- Sign prediction task (Section 4.4.5): In the Epinions and the Slashdot datasets, $\beta = 0.5$ and $\gamma = 0.8$. In the Wikipedia dataset, $\beta = 0.2$ and $\gamma = 0.6$.

4.4.2 Link Prediction Task

We evaluate the performance of personalized ranking models on link prediction in signed networks. The link prediction task is defined as follows: given a signed network and a seed node s, predict nodes which will be positively or negatively linked by the seed node in the future. An ideal personalized ranking for this task should place nodes that the seed node s potentially trusts (i.e., positive links) at the top, those that s potentially distrusts (i.e., negative links) at bottom, and other unknown ones in the



(a) GAUC on Epinions (b) GAUC on Slashdot (c) GAUC on Wikipedia

(d) AUC on Epinions (e) AUC on Slashdot (f) AUC on Wikipedia

Figure 4.7: **Link prediction performance of SRWR** in terms of GAUC and AUC. GAUC indicates how well a model ranks nodes to be positively connected by a seed node at the top and those to be negatively linked at the bottom. AUC indicates how many positive nodes are ranked higher than negative ones (see the details in Appendix A.4.1). Our proposed model SRWR shows the best link prediction performance in terms of GAUC and AUC for all the datasets.

middle. GAUC (Generalized AUC), proposed by [89], has been used to evaluate the quality of personalized rankings for link prediction in signed networks, and it measures such ideal ranking as 1.0. We also evaluate the ranking quality in terms of AUC indicating how many positive nodes are ranked higher than negative ones (see the details in Appendix A.4.1).

To perform this evaluation, we randomly select 1,000 seed nodes, and choose 20% edges of positive and negative edges from each seed node to form a validation set. Then, we randomly select another 1,000 seed nodes, and choose 20% edges of positive and negative edges from each seed node as a test set. We remove those se-

lected edges, and utilize the remaining edges as a training set to compute personalized rankings. For given a parameter combination, we measure GAUC on the personalized ranking w.r.t. each seed node in the validation set, and record the average GAUC over all the seed nodes. Then, we pick the best parameter combination that provides the highest average GAUC in the validation set. With the validated parameters, we report the average GAUC over all seed nodes in the test set. We perform the same procedure for AUC. For nodes directly connected with a seed node s in the training set, we exclude those nodes from a personalized ranking list w.r.t. s since we need to recommend links which are unknown to s.

Results. We compare SRWR to other random-walk based models M-RWR, M-PSALSA, PSR, RWR, and TR-TR on the link prediction task in signed networks. We also compare our method to GAUC-OPT which is a matrix factorization based link prediction method approximately maximizing GAUC [89]. As demonstrated in Figure 4.7, SRWR presents the best link prediction performance in terms of GAUC and AUC among the evaluated models over all the datasets. Compared to RWR which does not consider negative signs at all, our approach SRWR shows the significant improvement in the link prediction accuracy. Especially, GAUC of all other methods considering signed edges is higher than that of RWR as shown in Figure 4.7. This indicates that it is important to consider the sign of edges when we compute personalized rankings for link prediction in signed networks. Furthermore, SRWR outperforms other random walk based models including GAUC-OPT which is specially designed for this task, implying our signed surfer based on balance theory effectively estimates personalized rankings for link prediction in signed networks.

4.4.3 User Preference Preservation Task

Since a personalized ranking includes known and unknown users for a seed user (or node), how the ranking is consistent with the seed user's known preferences is also considered as one criterion for evaluating the quality of personalized rankings. In signed social networks, we consider that the known preferences of a seed user s are well preserved in a personalized ranking if positive users for s (i.e., they are positively connected by s) are at the top and negative ones are at the bottom in the ranking. Hence, an ideal ranking for s (excluding s from the ranking) should produce 1.0 GAUC with known positive and negative links from s in terms of user preference preservation. To evaluate the preference preservation performance of each method, we report the average GAUC over all test seed nodes without removing the selected test edges from a training set.

As shown in Table 4.4, our ranking model SRWR demonstrates the best GAUC in user preference preservation among all tested methods, indicating that SRWR almost perfectly preserves users' known preferences within their personalized rankings. The main reason for the result is that our signed surfer occasionally restarts at a seed node s with a positive sign; thus, the positive surfer frequently visits the positive neighbors of s, and the negative surfer frequently visits the negative neighbors of s. Hence, the trustworthiness scores on the positive neighbors are high, and those on the negative neighbors are low compared to those on nodes that are not connected by s.

One might think a simple approach that arbitrarily places the positive neighbors at the top, the negative ones at the bottom, and the other unknown nodes at the middle in a ranking list. The simple approach will produce 1.0 GAUC for user preference preservation; however, this cannot work on link prediction since we need to predict target nodes among unknown nodes (i.e., they are not connected to a seed node). On

Table 4.4: **User preference preservation quality of SRWR** in terms of GAUC (Appendix A.4.1). Note that 1.0 GAUC indicates that a method perfectly preserves a user's known preferences in its personalized ranking. Our proposed model SRWR shows the best performance in user preference preservation among all tested methods.

Datasets (GAUC)	**SRWR** (prop.)	**M-RWR**	**M-PSALSA**	**PSR**	**RWR**	**TR-TR**	**GAUC -OPT**
Epinions	**1.000**	0.999	0.902	0.730	0.708	0.650	0.824
Slashdot	**1.000**	0.982	0.800	0.728	0.705	0.625	0.708
Wikipedia	**0.999**	0.944	0.934	0.707	0.702	0.778	0.742

the contrary, our model SRWR is effective for not only user preference preservation but also signed link prediction as shown in Table 4.4 and Figure 4.7.

4.4.4 Troll Identification Task

In this section, we investigate the quality of a personalized ranking generated by SRWR in identifying trolls who behave abnormally or cause normal users to be irritated. The task is defined as follows: given a signed network and a normal user, identify trolls using a personalized ranking w.r.t. the user. In signed networks, we consider that a good personalized ranking of the normal user needs to capture trolls at the bottom of the ranking since most normal users are likely to dislike those trolls. Thus, we measure how well a personalized ranking of each method captures trolls at the bottom of the ranking to examine the quality of personalized node rankings.

As in the previous work [63], we also use the enemies of a user called *No-More-Trolls* in the Slashdot dataset as trolls. The user is an administrative account created for the purpose of collecting a troll list (i.e., the administrator is negatively connected to each troll). There are 96 trolls in the list. We exclude the edges adjacent to *No-More-Trolls* from the Slashdot dataset, and use the remaining edges to estimate a personalized ranking as a training set. We use the bottom-k of the ranking to search for those trolls. We randomly select $1,000$ seed nodes as a validation set to search

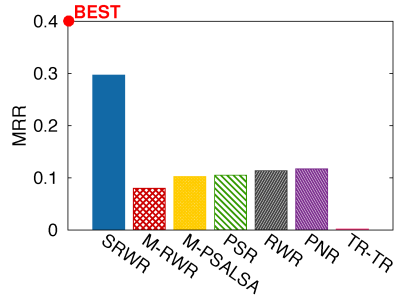
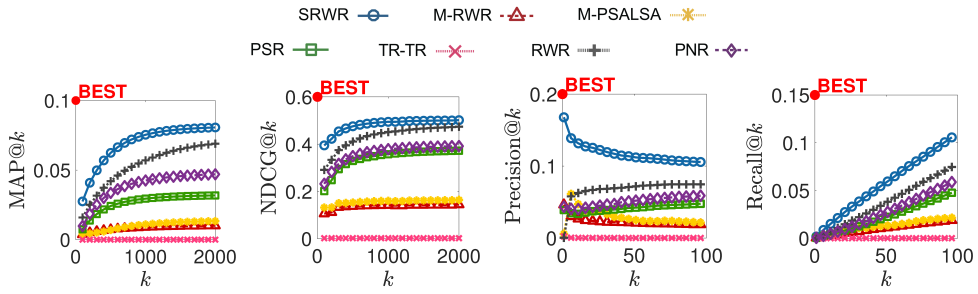


Figure 4.8: **Performance on troll identification of SRWR w.r.t. MRR.** The measure indicates how trolls are ranked low in a personalized ranking. The SRWR is the highest MRR among all tested models.



(a) MAP@k (b) NDCG@k (c) Precision@k (d) Recall@k

Figure 4.9: **Performance of SRWR for the troll identification task through various measurements**: MAP@k (4.9(a)), NDCG@k (4.9(b)), Precision@k (4.9(c)), and Recall@k (4.9(d)). SRWR shows the best performance for all the measurements compared to other competitors.

for hyper-parameters required by each method. We pick the best parameter combination that provides the highest Mean Reciprocal Rank (MRR) in the validation set. Then, for each user, we search for trolls within the bottom-k ranking, and evaluate how those trolls are ranked low in the ranking, which is measured by MRR. We also measure Mean Average Precision (MAP@k), Normalized Discount Cumulated Gain (NDCG@k), Precision@k, and Recall@k to check the performance of each method in terms of various metrics (see the details in Appendix A.4.2). Since there are no user-graded scores for the troll list, we set those scores to 1 for NDCG.

Results. Our proposed model SRWR significantly outperforms other ranking models for the troll identification task as shown in Figures 4.8 and 4.9. According to

Figure 4.8, the rank of a bottom ranked troll from our model is lower than that of other ranking models because MRR of our model is the highest compared to other competitors. More trolls are captured within the bottom-k ranking produced by our proposed model according to MAP@k shown in Figure 4.9(a). Note that Figures 4.9(c) and 4.9(d) indicate that SRWR achieves higher Precision@k and Recall@k for capturing trolls than other methods. SRWR provides $4\times$ better performance than PNR, the second best one, in terms of Precision@k when $k = 1$. Many trolls tend to be ranked low in our personalized ranking because SRWR achieves better MAP@k and NDCG@k than other ranking models as presented in Figures 4.9(a) and 4.9(b).

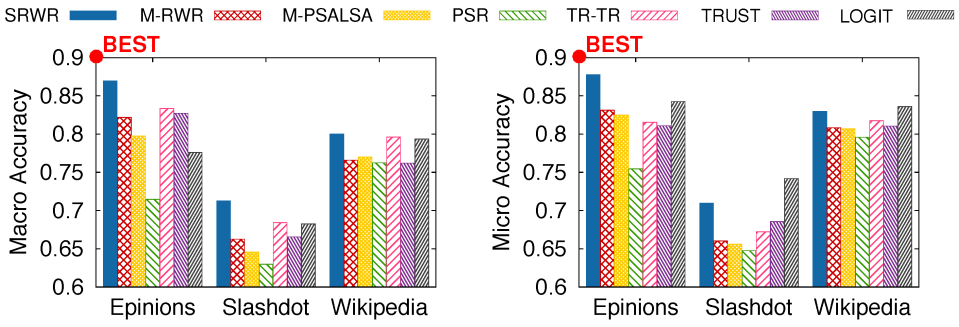
Case study. We investigate the top-20 and the bottom-20 of the personalized ranking for a user called *"yagu"* in Table 4.5. We list the users in the bottom-20 ranking in the ascending order of the ranking scores in Table 4.5. According to the result, more trolls are ranked low in the personalized ranking from SRWR, indicating that our model is more sensitive in capturing trolls than other models. Also, the query user is ranked low at the bottom of the ranking from M-PSALSA while the user is ranked high in the ranking from our model. The query user should trust himself; thus, the user should be ranked at the top in a personalized ranking. This implies our model is more desirable than other models for personalized rankings in signed networks.

Table 4.5: **Troll prediction results** of ranking models w.r.t. a normal user *"yagu"*. For each model, we show top-20 (trusted) and bottom-20 (distrusted) nodes based on the personalized ranking for "yagu". The users in the bottom-20 ranking are sorted in the ascending order of the ranking scores. Red-colored users (†) are trolls, a blue-colored user (⋆) is a query user, and the black-colored (non-marked) are normal users. Note that SRWR shows the best result: in SRWR, the query user is ranked 1st, and many trolls are ranked at the bottom in the personalized ranking. M-PSALSA provides inferior results since they rank the query user high at the bottom of the ranking, although the query user is the most trusted user for this task. M-RWR, PSR and TR-TR are not satisfactory either: they do not capture many trolls at the bottom of their rankings.

	SRWR (proposed)		**M-RWR**		**M-PSALSA**		**PSR**		**TR-TR**	
Rank	**Trust Ranking**	**Distrust Ranking**	**Trust Ranking**	**Distrust Ranking**	**Trust Ranking**	**Distrust Ranking**	**Trust Ranking**	**Distrust Ranking**	**Trust Ranking**	**Distrust Ranking**
1	**yagu**⋆	**Klerck**†	**yagu**⋆	dubba-d	Work+Ac	HanzoSa	**yagu**⋆	SmurfBu	**yagu**⋆	Jack+B.
2	Photon+	**Adolf+H**†	Bruce+P	derago	Unknown	**Jerk+Ci**†	Uruk	Dr.Seus	dexterp	inTheLo
3	Uruk	GISGEOL	CmdrTac	msfodde	afidel	NineNin	Photon+	Doctor_	Jamie+Z	Mactrop
4	stukton	Nimrang	CleverN	cramus	heirony	Rogerbo	clump	artoo	ryanr	DiceMe
5	TTMuskr	Kafka_C	Uruk	lakerdo	bokmann	**SexyKel**†	TTMuskr	Juggle	KshGodd	Einstei
6	clump	Thinkit	Photon+	p414din	ezeri	ScottKi	stukton	FreakyG	TheIndi	FinchWo
7	Bruce+P	**CmderTa**†	stukton	an+unor	As+Seen	qurob	RxScram	RunFatB	daoine	Penus+T
8	RxScram	SteakNS	clump	exfuga	KillerD	bendodg	charlie	jmpoast	Berylli	r_glen
9	CmdrTac	JonKatz	TTMuskr	kryptok	potaz	ArnoldY	ssbg	El_Muer	danhara	Roland+
10	aphor	Henry+V	RxScram	toomz	byolinu	jcr	Idarubi	Ghost+H	Degrees	sting3r
11	CleverN	Miguel+	John+Ca	Shazzma	Stanist	davesch	spotted	The+Hob	charlie	Tuvai
12	throx	ringbar	aphor	Jetboy0	TripMas	List+of	Golias	bananac	sagei	1234567
13	chrisd	fimbulv	chrisd	KrisCow	andy+la	**yagu**⋆	Slider4	Sodade	tadghin	1337_h4
14	CowboyN	by+Fort	TripMas	%2BMaje	bani	linzeal	Twid	peattle	capocci	1g%24ma
15	Blakey+	**I+Am+Th**†	kfg	frankyf	SETIGuy	jeffy12	Toast	GISGEOL	einstei	%2B%2Bg
16	Hemos	VAXGeek	Hemos	Lukano	generic	foobar1	Unknown	Nimrang	jebell	3p1ph4n
17	cgenman	NineNin	davesch	J'raxis	Bi()haz	dealsit	nanojat	**Adolf+H**†	pythorl	4d49434
18	TripMas	dfenstr	CowboyN	funklor	_xeno_	UbuntuD	lucasth	Kafka_C	sphere	4e61747
19	kfg	Quantum	NewYork	staynz7	HeyLaug	greenrd	pitboss	SteakNS	winneto	ABeowul
20	topham	SmurfBu	dada21	ikkibr	Eli+Got	Concern	harlows	Thinkit	Allen+V	Absolut

⋆ This indicates the querying normal user.

† This indicates a troll.



(a) Macro accuracy (b) Micro accuracy

Figure 4.10: **Performance of SRWR on sign prediction** in terms of macro and micro accuracies where the macro accuracy indicates the average seed-wise accuracy, and the micro accuracy indicates the ratio of the number of correct predictions to the total test edges. While the micro accuracy of SRWR is the second best, the macro accuracy of SRWR is the best compared to its competitors.

4.4.5 Sign Prediction Task

We evaluate ranking scores produced by each ranking model rather than the order between nodes. Note that a ranking score between a seed node s and a target node t is based on the trustworthiness between those nodes. Hence, it is also important to examine how well those ranking scores reflect trust relationships between nodes. We measure the quality of those ranking scores exploiting the sign prediction task which is defined as follows: given a signed network and a seed node s where signs of edges connected from s are missed, predict those signs using the personalized ranking scores of each method with respect to the seed node s.

To construct a validation set, we randomly select $1,000$ seed nodes, and choose 20% edges of positive and negative links from each seed node. We also randomly select another $1,000$ seed nodes, and choose 20% positive and negative edges from each seed node to form a test set. Then, we remove each selected edge $(s \to t)$, and predict the edge's sign based on personalized ranking scores w.r.t. node s in the graph

represented by the remaining edges. Our ranking score vector is $\mathbf{r} = \mathbf{r}^+ - \mathbf{r}^-$ whose values range from -1 to 1. If $\mathbf{r}_t$ is greater than or equal to 0, then we predict the sign of the edge $(s \to t)$ as positive. Otherwise, it is considered as negative. We pick the best parameter combination having the highest micro accuracy (see the below) in the validation set. With the validated parameters, we measure the following prediction accuracies of a test set, *macro and micro accuracies* which are defined as follows:

$$\text{macro accuracy} = \frac{1}{n_Q} \sum_{i=1}^{n_Q} \text{accuracy}(i)$$

$$\text{micro accuracy} = \frac{\text{\# of correct predictions}}{\text{\# of total test edges}}$$

where n_Q is the number of test seed nodes, and accuracy(i) is the seed-wise accuracy of i-th test seed node (i.e., the ratio of the number of correct predictions to the number of test edges on i-th seed node).

Results. We compare the performance of SRWR to that of other random walk based ranking models M-RWR, M-PSALSA, TR-TR, and PSR on the sign prediction task. We also compare our model SRWR to TRUST [91] and LOGIT [90] which are specially designed for predicting signs between two arbitrary nodes in signed networks. As shown in Figure 4.10(a), SRWR shows the best macro accuracy among all tested methods. Although SRWR obtains higher micro accuracy than LOGIT in the Epinions dataset, the micro accuracy of LOGIT is better than that of SRWR in other datasets as shown in Figure 4.10(b).

Another observation is that LOGIT has a large gap between macro and micro accuracies while SRWR has a small gap as shown in Figure 4.10. A large gap implies that on average, the deviation between micro accuracy and seed-wise accuracy (i.e., accuracy(i)) is large, i.e., accuracy(i) for i-th test seed node is likely to deviate sub-

Table 4.6: **Difference between SRWR and LOGIT on sign prediction** in terms of macro accuracies of high and low degree groups. The overall group is the union of the high and low degree groups. We measure the average of seed-wise accuracies for each group (i.e., the macro accuracy of the group) and the standard deviation between accuracies of those groups. LOGIT tends to predict a seed node's test edges in the high degree group better than SRWR, while SRWR predicts better those in the low degree group compared to LOGIT. Also, the result on the standard deviation indicates that the disparity of SRWR between accuracies of those groups is smaller than that of LOGIT.

Datasets	Methods	Overall Group	High Degree Group	Low Degree Group	Standard Deviation
Epinions	SRWR	**0.8696**	**0.8876**	**0.8651**	**0.0159**
	LOGIT	0.7762	0.8760	0.7510	0.0883
Slashdot	SRWR	**0.7128**	0.7133	**0.7127**	**0.0004**
	LOGIT	0.6827	**0.7943**	0.6546	0.0987
Wikipedia	SRWR	**0.8004**	0.8556	**0.7865**	**0.0489**
	LOGIT	0.7937	**0.8671**	0.7752	0.0650

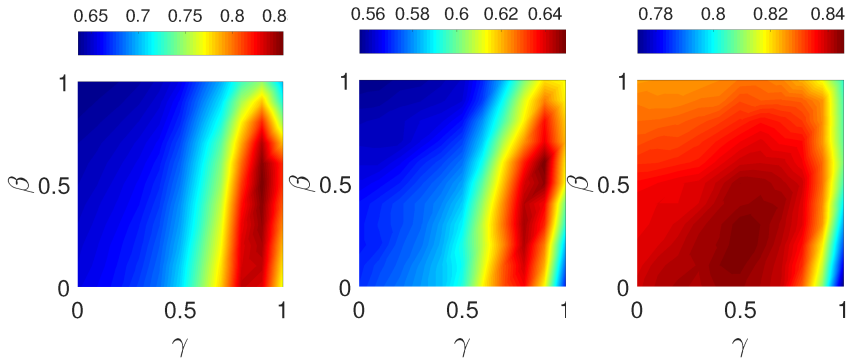
stantially from the micro accuracy. To analyze such deviation, we look into seed-wise accuracies in terms of node degrees. Since there are a few high degree nodes and a lot of low degree nodes in real-world graphs according to power-law degree distribution [119], we split test seed nodes into two groups as follows: high (top-20%) and low (bottom-80%) groups in the order of out-degrees of test seed nodes. Then, we measure the average of seed-wise accuracies for each group (i.e., the macro accuracy of the group) and the standard deviation between the accuracies of those groups.

According to Table 4.6, LOGIT tends to better predict test edges of a seed node in the high degree group than those in the low degree one. In particular, on the Epinions and the Slashdot datasets, the macro accuracy of LOGIT in the low degree group is rather lower than that of LOGIT in the high degree group. These results imply that LOGIT is biased toward predicting test edges from a high degree seed node. Note that the number of test edges from a high degree node is larger than that of test edges from a low degree node since 20% test edges are randomly extracted from each selected

test node. Thus, the total number of correct predictions from LOGIT is large (i.e., the micro accuracy becomes high). However, the seed-wise accuracies of LOGIT are low in the low degree group (i.e., the macro accuracy becomes low) as shown in Table 4.6, thereby increasing the gap of LOGIT between micro and macro accuracies.

On the contrary, the gap of SRWR between micro and macro accuracies are relatively smaller than that of LOGIT as shown in Figure 4.10, along with the small standard deviation of SRWR as shown in Table 4.6. Note that SRWR outperforms LOGIT in the low degree group over all the datasets as shown in Table 4.6. That is why the macro accuracy of SRWR is higher than that of LOGIT for the total test seed nodes as shown in Figure 4.10. SRWR also shows a satisfactory performance in the high degree group, especially on the Epinions dataset, although the performance of SRWR is not better than that of LOGIT on the Slashdot and the Wikipedia datasets as shown in Table 4.6. Thus the standard deviation of SRWR between those groups is smaller than that of LOGIT. These experimental results indicate that SRWR is competitive enough to be comparable to other models such as LOGIT in the sign prediction task.

Note that LOGIT is a graph feature based method which exploits local graph features, within 1 hop from seed and target nodes, such as node degrees, common neighbors, and local wedges for predicting the sign between the seed and target nodes. A high degree node is likely to have plentiful features, since the high degree node has many connections to other nodes. A low degree node would not have such local features enough due to less connections; hence, LOGIT has a limitation on increasing the predictive performance for test edges from the low degree node based only on local graph features. On the other hand, SRWR's inference is based on the information more than 1 hop from the seed node because the signed random surfer visits the target node via various length of paths from the seed node to the target node. That

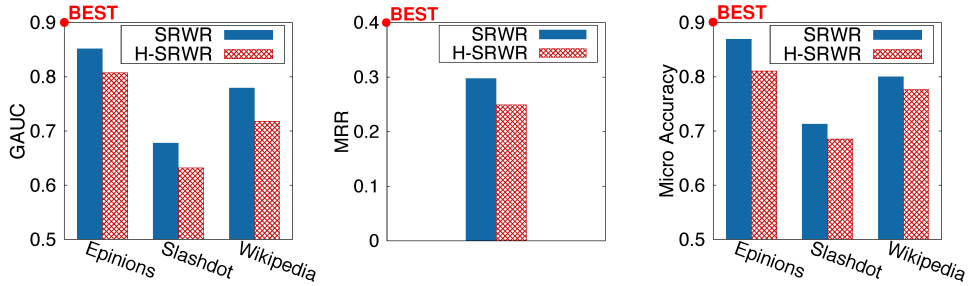


(a) Epinions (b) Slashdot (c) Wikipedia

Figure 4.11: **Accuracy maps of SRWR according to balance attenuation factors β and γ** where each color indicates a degree of accuracy. The Epinions and the Slashdot datasets present similar tendencies while the Wikipedia dataset shows a different result from those of the two datasets.

is why SRWR works well on predicting test edges of low degree nodes compared to LOGIT.

Balance attenuation factors. We adjust the balance attenuation factors of SRWR, and evaluate the sign prediction task in terms of micro accuracy to examine how well balance theory explains signed networks. In this experiment, we use the top-100 highest degree nodes as a test set for each network. The Epinions and the Slashdot datasets show similar results where larger values of β and γ achieve high accuracy as shown in Figures 4.11(a) and 4.11(b). Unlike these two datasets, the accuracy is high when β is small in the Wikipedia network as shown in Figure 4.11(c). This implies that "an enemy of my enemy is my friend" would not be correct in the network, which means balance theory does not apply well to the Wikipedia dataset. The reason is that the Wikipedia network represents votes between users to elect administrators; thus, the dataset is different from the Epinions and the Slashdot networks which are general social networks. Note that the validated balance attenuation factors for the sign prediction task in Section 4.4.1 are consistent with the tendency



(a) GAUC of the link prediction task

(b) MRR of the troll identification task

(c) Micro Accuracy of the sign prediction task

Figure 4.12: **Effect of the balance attenuation factors of SRWR.** The performance of SRWR is better than that of H-SRWR (i.e., SRWR without using balance attenuation factors) in terms of the link prediction, the troll identification, and the sign prediction tasks.

demonstrated in Figure 4.11. Another observation is that the ideal balance theory does not apply to real-world signed networks because the accuracy is not the best over all datasets when $\beta = 1$ and $\gamma = 1$ (i.e., the ideal balance theory).

4.4.6 Effectiveness of Balance Attenuation Factors

We examine the effects of the balance attenuation factors of SRWR on the performance of the link prediction, the troll identification, and the sign prediction tasks. In this experiment, we use H-SRWR ($\beta = 1$ and $\gamma = 1$) and SRWR with validated balance factors for each dataset as mentioned in Section 4.4.1. H-SRWR indicates that we compute SRWR scores using Equation (4.2) which does not adopt balance attenuation factors. We measure GAUC for link prediction, MRR for troll prediction, and micro accuracy for sign prediction to compare SRWR and H-SRWR.

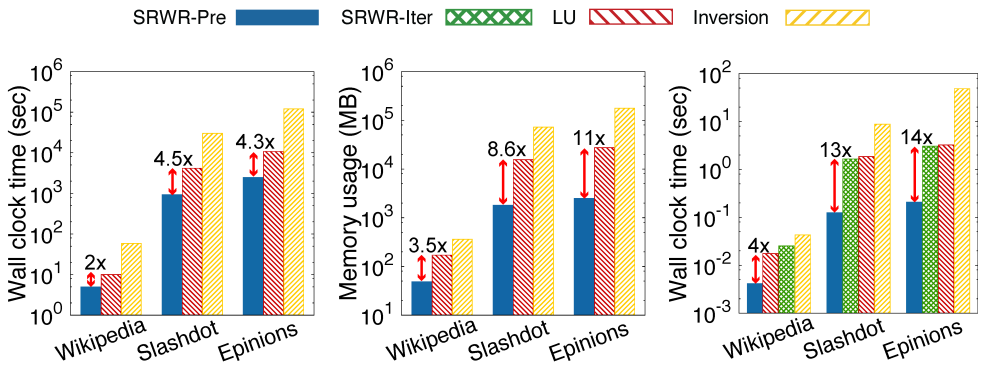
Figure 4.12 indicates that introducing balance attenuation factors is helpful for improving the performance of each application in signed networks. As shown in Figure 4.12(a), SRWR obtains higher GAUC than H-SRWR in the link prediction task. Also, Figure 4.12(b) presents that SRWR achieves better MRR than H-SRWR on the

troll identification task. Moreover, the accuracy of SRWR is higher than that of H-SRWR over all datasets for the sign prediction task as presented in Figure 4.12(c). Although introducing balance attenuation factors increase the complexity of our model and demand an additional step for searching those factors, it makes our model flexible so that SRWR resolves the weakness inherent from the strong balance theory as discussed in Section 4.3.1.2 through adjusting those factors, and improves the performance of each application in signed social networks.

4.4.7 Performance of SRWR-Pre

We investigate the performance of our preprocessing method SRWR-Pre in terms of preprocessing time, memory space for precomputed data, and query time. We compare SRWR-Pre to other baseline preprocessing methods Inversion and LU as well as our iterative method SRWR-Iter. Preprocessing and query time are measured in wall-clock time, and we measure the average query time for $1,000$ random seed nodes. We set $\beta = 0.5$, $\gamma = 0.5$, $c = 0.05$ for all tested methods. In SRWR-Pre, we set the hub selection ratio $k = 0.001$ for the hub-and-spoke reordering method to make the number of hubs n_2 small enough as in [54]. We also measure how much memory space each preprocessing method needs for the precomputed matrices to compare memory efficiency. We omit bars for SRWR-Iter in Figures 4.13(a) and 4.13(b) because SRWR-Iter does not involve a heavy preprocessing phase (i.e., the time cost for the normalization phase of SRWR-Iter in Algorithm 5 is trivial, and the memory usage of SRWR-Iter is equal to that of the input graph).

Figures 4.13(a) and 4.13(b) show that SRWR-Pre provides better performance than LU and Inversion in terms of preprocessing time and memory space for preprocessed data. SRWR-Pre is up to $4.5\times$ faster than the second best preprocess-



(a) Preprocessing time (b) Memory space for preprocessed data (c) Query time

Figure 4.13: **Performance of SRWR-Pre**: (a) and (b) show the comparison of the preprocessing time and the memory space for preprocessed data among preprocessing methods; (c) compares the query time among all tested methods. SRWR-Pre presents the best performance compared to other preprocessing methods in terms of preprocessing time and memory efficiency. SRWR-Pre also computes SRWR scores more quickly than SRWR-Iter and the baseline methods.

Table 4.7: **Total number of non-zeros (nnz_t) in precomputed matrices for each preprocessing method.** Our method SRWR-Pre generates less non-zeros in precomputed matrices than other preprocessing methods.

Dataset	A: nnz_t in SRWR-Pre	B: nnz_t in LU	C: nnz_t in Inversion	Ratio B/A	Ratio C/A
Wikipedia	3,207,758	11,257,644	23,928,232	3.51	7.46
Slashdot	119,580,272	1,032,276,955	4,817,461,830	8.63	40.29
Epinions	165,006,379	1,825,755,902	11,755,245,476	11.06	71.24

ing method LU in terms of preprocessing time. Also, SRWR-Pre requires up to 11× less memory space than LU. Especially, our method SRWR-Pre uses 2.6GB memory for the precomputed data in the Epinions dataset while LU and Inversion require 28GB and 180GB memory, respectively. These results imply that SRWR-Pre is fast and memory-efficient compared to other preprocessing methods. SRWR-Pre also shows the fastest query speed among other competitors including our iterative method SRWR-Iter as presented in Figure 4.13(c). SRWR-Pre is up to 14× faster than SRWR-Iter, and up to 15× faster than the second best preprocessing method LU in

the Epinions dataset. Note that SRWR-Pre computes SRWR scores for a given seed node in less than 0.3 second over all signed networks. Inversion is the slowest among the tested methods over all datasets. The main reason is that Inversion produces a very large number of non-zeros in precomputed matrices (e.g., Inversion produces about 11 billion non-zeros in the Epinions dataset as presented in Table 4.7). These results indicate that SRWR-Pre is appropriate to serve given queries in real-time on the datasets with low memory usage compared to other methods.

Discussion. In this work, we propose two methods for SRWR: SRWR-Iter and SRWR-Pre which are iterative and preprocessing methods computing SRWR scores, respectively. SRWR-Iter does not require heavy precomputed data to compute SRWR scores. However, SRWR-Iter shows slow query speed as presented in Figure 4.13(c) because SRWR-Iter should perform matrix vector multiplications many times for a given seed node. On the other hand, SRWR-Pre is faster up to 14× than SRWR-Iter in term of query speed since SRWR-Pre directly computes SRWR scores from precomputed data. However, in SRWR-Pre, the values of the parameters c, β, and γ of SRWR are fixed through the preprocessing phase (Algorithm 7); thus, SRWR-Pre cannot change the parameters in the query phase (Algorithm 8). To obtain SRWR scores with the different values of the parameters, we need to perform the preprocessing phase with the parameters again. On the contrary, SRWR-Iter easily handles the change of the parameters in the query phase (Algorithm 6) without additional operations such as preprocessing. One appropriate usage for our methods is that a user uses SRWR-Iter to find proper parameters for a specific application; and then, the user exploits SRWR-Pre with the discovered parameters to accelerate the query speed in the application.

4.5 Summary

We propose Signed Random Walk with Restart, a novel model which provides personalized trust or distrust rankings in signed networks. In our model, we introduce a signed random surfer so that she considers negative edges by changing her sign for surfing on signed networks. Consequently, our model provides personalized trust or distrust rankings reflecting signed edges. Our model is a generalized version of Random Walk with Restart working on both signed and unsigned networks. We also devise SRWR-Iter and SRWR-Pre, iterative and preprocessing methods to compute SRWR scores, respectively. We experimentally show that SRWR achieves the best accuracy for link prediction, predicts trolls $4\times$ more accurately, and shows a satisfactory performance for inferring missing signs of edges compared to other methods. SRWR-Pre preprocesses a signed network up to $4.5\times$ faster, and requires $11\times$ less memory space than other preprocessing methods; SRWR-Pre computes SRWR scores up to $14\times$ faster than other methods. Future research directions include developing a learning algorithm which automatically learns the balance attenuation factors of our model from a given input graph.

Chapter 5

Relational Reasoning in Edge-labeled Graphs

5.1 Introduction

How can we accurately infer the relation between two arbitrary nodes in edge-labeled graphs? Understanding how nodes are related is crucial for analyzing graph data, and many researchers have designed various relevance measures to effectively identify relevance between nodes in graphs. Random Walk with Restart (RWR) [56], a random surfer model, has been utilized for measuring relevance scores between nodes with considering *global network structure* [19] and *multi-faceted paths* between nodes [20]. RWR has been extensively utilized in numerous graph mining applications such as personalized ranking [56], link prediction [30], recommender systems [32], anomaly detection [120], community detection [47], etc.

Although many networks have been modeled with multiple edge labels to represent diverse relationships between nodes [121], RWR has a limitation in inferring the edge's label between two nodes in edge-labeled graphs. For example, social networks such as Slashdot [63] represent trust or distrust for users as positive or negative edges. In knowledge graphs such as WordNet [122], concepts are associated with predicates. Since RWR does not consider edge labels for its relevance, it cannot identify how nodes are related with in terms of edge labels (see Figure 5.1). For relation inference on multiple edge labels, there are two main approaches: path feature

models such as Path Ranking Algorithm (PRA) [67] and translation models such as TransE [92] and TransR [93]. PRA uses a surfer to exploit paths between two nodes as features for predicting their relation. However, PRA's reasoning focuses on relatively short paths due to expensive path enumerations. On the other hand, TransE and TransR discover latent embeddings for relations and entities under a relational translation scenario; however, they do not take account of paths between nodes into those embeddings. These limitations make the reasoning of those approaches miss the information provided by complex multi-hop paths between the nodes.

In this work, we propose MuRWR (Multi-Labeled Random Walk with Restart), a novel random surfer model which accurately identifies label relevance between two nodes in an edge-labeled graph (see Figure 5.2). Our approaches are 1) to introduce a *labeled random surfer* whose label indicates the relation between starting and visiting nodes, 2) to change the surfer's label during random walks for multi-hop reasoning process, and 3) to learn suitable rules on how to change her label from the given edge-labeled graph. We show that letting the labeled surfer move around the graph enables our model to do accurate multi-hop relational reasoning without explicit path enumeration, as well as to generalize RWR into edge-labeled graphs. Through extensive experiments, we show that MuRWR predicts edge labels between nodes more accurately than existing models. Our main contributions are as follows:

- **Model.** We propose MuRWR (Multi-Labeled Random Walk with Restart), a novel and accurate model for relation inference in edge-labeled graphs (Definition 5.3). MuRWR exploits a labeled random surfer who considers edge labels to compute effective relevance scores between nodes for each edge label. We show that MuRWR is a generalized version of RWR to edge-labeled graphs (Property 4).

- **Algorithm.** We propose how to learn the surfer's label from an edge-labeled graph (Lemma 5.1) and an iterative algorithm for MuRWR (Algorithm 11). We also theoretically prove the convergence of the algorithm (Theorem 5.1).
- **Experiment.** Our experiments show MuRWR provides the most accurate performance for relation inference (Tables 5.3 and 5.4).

The rest of the paper is organized as follows. We provide preliminaries on a traditional random walk based model, and formally define edge-labeled graphs in Section 5.2. We describe MuRWR and algorithms for computing MuRWR scores in Section 5.3. After presenting experimental results in Section 5.5, we summarize this work in Section 5.6. The symbols used in this paper are in Table 5.1.

5.2 Preliminary

We provide preliminaries on relation inference in edge-labeled graphs, our target research task, and RWR in this section. Note that we use the terms *edge label* and *relation* interchangeably.

Edge-labeled Graphs. Many researchers from various application areas are confronted with labeled edges which encode diverse relations between entities in graphs [63, 122, 121, 123]. We follow the formal notations for an edge-labeled graph, defined in Definition 2.3 of Chapter 2.

Relation Inference Task. We describe the formal problem definition of relation inference handled in this paper.

Problem 2 (Relation Inference [67])**.** *The relation inference task is defined as follows:*

- ***Given**: an edge-labeled graph $G = (\mathbf{V}, \mathbf{E}, \mathbf{L})$, and two disconnected nodes s and t,*
- ***Infer**: the edge label from source node s to target node t among edge labels in $\mathbf{L}$.*

Table 5.1: Table of symbols used in Chapter 5.

Symbol	Definition
G	input edge-labeled graph
n	number of nodes in G
m	number of edges in G
K	number of edge labels in G
s	source node
c	restart probability
l_i	i-th edge label where $1 \leq i \leq K$
l_{uv}	label on the edge from node u to node v
$\overleftarrow{\mathbf{N}}_v$	set of in-neighbors of node v
$\overrightarrow{\mathbf{N}}_v$	set of out-neighbors of node v
$\mathbf{L}$	set of edge labels, i.e., $\mathbf{L} = \{l_1, \cdots, l_K\}$
$\mathbf{A}$	$(n \times n)$ labeled adjacency matrix of G
$\mathbf{A}_k$	$(n \times n)$ l_k-labeled semi-adjacency matrix
$\mathbf{A}_{-k}$	$(n \times n)$ l_k-labeled semi-row-normalized matrix
$\mathbf{R}$	$(n \times K)$ relevance score matrix w.r.t. source node s
$\mathbf{R}_{vi}$	relevance score between nodes s and v for edge label l_i
$\mathbf{S}_k$	$(K \times K)$ label transition probability matrix on l_k
$\mathbf{S}_{kij}$	probability that the surfer's label changes from l_i to l_j through l_k-labeled edge, i.e., $P(l_i \xrightarrow{l_k} l_j)$
$\mathbf{1}(\cdot)$	indicator function that returns 1 if a given predicate is true, or 0 otherwise

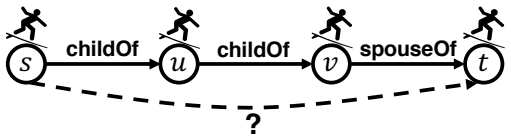


Figure 5.1: **Limitation of a random surfer in traditional RWR** for estimating relation of s and t. Since the surfer does not consider edge labels, it cannot identify the relation.

Note that Problem 2 is different from that of belief propagation (BP) [124, 125] that aims to classify labels on nodes not edges while the goal of MuRWR is to predict labels on edges.

The main challenge is to devise how to reflect multiple edge labels on relevance scores between nodes in edge-labeled graphs. Although RWR is able to identify relevance scores between nodes, it depends only on the link structure of the graph without considering labels; therefore, it cannot estimate how nodes are related in terms of edge labels. Figure 5.1 shows an example of representing family relationships such

as childOf and spouseOf. The traditional surfer in RWR does not consider edge labels at all during its walks while the labeled edges should be interpreted differently. Thus, the information encoded in the path is ignored when measuring the relevance between nodes s and t; RWR cannot identify which edge label should be associated with nodes s and t. How can we make the surfer recognize edge labels to correctly predict their relation?

5.3 Proposed Method

We propose MuRWR (Multi-Labeled Random Walk with Restart), a novel random walk based model which measures relevance scores between a source node and other nodes for each edge label in an edge-labeled graph. The technical challenges and our main ideas are as follows:

- How can we make a surfer consider the edge labels? We introduce a *labeled random surfer* whose label at a node indicates the relation from the source node to that node.
- How can the surfer infer the relation between the nodes? We allow the surfer to change her label during random walks with *rules* for a multi-hop reasoning.
- How can we discover the rules? We exploit a data-driven approach to extract knowledge from the graph so that the surfer learns the rules.

In MuRWR, the surfer's movement from source node s to a node t is considered as a process of reasoning about the relation from s to t. Figure 5.2 depicts how MuRWR infers the relation from node s to node t. The labeled surfer has one of edge-labels such as childOf (blue-colored) or grandchildOf (red-colored) at each node except at s. Assume we have the following rules for the surfer:

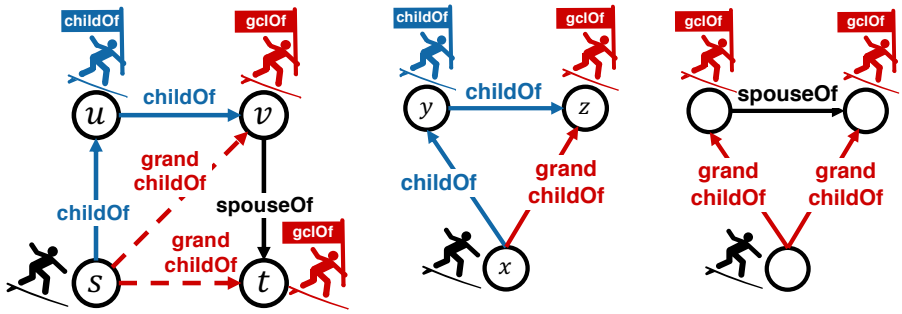


Figure 5.2: **Examples of labeled walks and label transitive triangles.** (a) shows how our labeled surfer walks along the path from s to t. The blue surfer is for childOf, the red one is for gclOf (grandchildOf), and the black one is unlabeled. In (a), a dashed line indicates two disconnected nodes are related with a label inferred by the surfer. (b) and (c) are the examples of label transitive triangles used for learning the rules on how to change the surfer's label.

Rule 1. If childOf-surfer moves along childOf-edge, her label changes to grandchildOf.

Rule 2. If grandchildOf-surfer moves along spouseOf-edge, her label remains the same, i.e., grandchildOf.

In Figure 5.2(a), the surfer first starts from source node s without any label, and moves to node u. Then, she has childOf label at u since nodes s and u are directly connected with childOf-edge (note that her label indicates the relation between the *source* node s and the current node). After she moves to node v, her label changes to grandchildOf by the first rule. At this time, MuRWR infers the relation from s to v as grandchildOf as shown in Figure 5.2(a). When the surfer finally arrives at node t, her label is still grandchildOf by the second rule, indicating that the relation from s to t is also inferred as grandchildOf. Thus, introducing a label to the surfer enables her to do a multi-hop relational reasoning by walking around the graph if the surfer knows appropriate rules.

Then, how can we discover the rules for the surfer? For this, we exploit a data-

driven approach, which extracts knowledge embraced in the given graph. The knowledge that the surfer learns is represented as labeled transitive triangles as in Figures 5.2(b) and 5.2(c). A transitive triangle is interpreted as a syllogistic knowledge, e.g., Figure 5.2(b) implies

$$\underbrace{\text{childOf}\ (x, y)}_{\text{current surfer's label}} \wedge \underbrace{\text{childOf}\ (y, z)}_{\text{edge label}} \Rightarrow \underbrace{\text{grandchildOf}\ (x, z)}_{\text{next surfer's label}}.$$

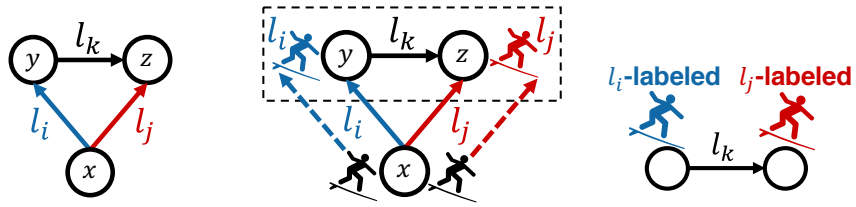
An intuitive example for the surfer's learning is as follows. Suppose an untrained surfer (black-colored) is at node x in Figure 5.2(b). The surfer then has childOf-label at node y since x and y are directly related with the label. Similarly, she has grandchildOf-label at node z. This is interpreted as: if childOf-surfer moves along childOf-edge, her label changes to grandchildOf. Thus, we enumerate transitive triangles from the graph, and use them as training instances, called *label transition observations*, for the surfer.

We first describe the details on label translation observations in Section 5.3.1, and explain how to learn rules for our labeled surfer from the observations in Section 5.3.2. Then, we formally define and formulate our model MuRWR in Sections 5.3.3 and 5.3.4, respectively. We present the algorithms for computing MuRWR in Section 5.3.5, and prove its convergence.

5.3.1 Label Transition Observation

We describe how to collect observations for learning the surfer's rules from the graph G. We first define *label transition observation* as follows:

Definition 5.1 (Label Transition Observation)**.** *A label transition observation $l_i \xrightarrow{l_k} l_j$ is that when l_i-labeled surfer moves along l_k-labeled edge, her label changes to l_j.* ■



(a) Label transitive relationship

(b) Interpretation of the transitive triangle

(c) Label transition observation

Figure 5.3: **Example of how to obtain label transition observations from label transitive relationships.** (a) presents a label transitive relationship. (b) shows how to interpret the triangle to obtain the label transition observation $l_i \xrightarrow{l_k} l_j$ in (c).

A label transition observation is obtained from a *label transitive triangle* interpreted as a transition between edge labels. For example, suppose an untrained surfer (black-colored) is at node x in Figure 5.3(a). When the surfer moves to node y, her label should be l_i because x and y are directly related with label l_i. Similarly, her label is l_j at node z. Then we observe how the surfer's label changes as in Figure 5.3(c): when l_i-labeled surfer moves along l_k-labeled edge, her label changes to l_j, implying the label transition observation $l_i \xrightarrow{l_k} l_j$ in Definition 5.1. To collect such observations, we enumerate all transitive triangles from the graph G using an efficient triangle enumeration algorithm [126].

5.3.2 Learning Label Transition Probabilities

We explain how to learn the surfer's rules from the label transition observations. A *label transition probability* is a conditional probability $P(l_j|l_i, l_k)$ such that when l_i-labeled surfer moves along l_k-labeled edge, her label changes to l_j. For brevity, let $P(l_i \xrightarrow{l_k} l_j)$ denote $P(l_j|l_i, l_k)$. We formally define *label transition probability matrix* as follows:

Definition 5.2 (Label Transition Probability Matrix)**.** *Let* $\mathbf{S}_k \in \mathbb{R}^{K\times K}$ *be a label transition probability matrix on edge label* l_k*. The* (i, j)*-th entry* $\mathbf{S}_{kij}$ *of* $\mathbf{S}_k$ *indicates the label*

transition probability that the surfer's label changes from l_i to l_j through edge label l_k, i.e., $\mathbf{S}_{kij} = P(l_i \xrightarrow{l_k} l_j)$. *Note that* $\sum_{j=1}^{K} P(l_j|l_i,l_k) = \sum_{j=1}^{K} \mathbf{S}_{kij} = 1$. ■

Given label transition observations, we aim to learn label transition probabilities $\mathbf{S}_{kij}$ maximizing a likelihood function of making the observations. Suppose we are given sets $\mathcal{D}_k$ of label transition observations represented as follows:

$$\mathcal{D}_k = \{\mathbf{x}_{kh} = (\mathbf{x}_{khs} \xrightarrow{l_k} \mathbf{x}_{kht}) | 1 \leq h \leq n_k\} \text{ for } 1 \leq k \leq K$$

where n_k is the number of the observations in $\mathcal{D}_k$. $\mathbf{x}_{kh}$ is a label transition observation $\mathbf{x}_{khs} \xrightarrow{l_k} \mathbf{x}_{kht}$ such that $\mathbf{x}_{khs}, \mathbf{x}_{kht} \in \mathbf{L}$ where $\mathbf{x}_{khs}$ is the surfer's source label, and $\mathbf{x}_{kht}$ is her destination label. Let $P(\mathbf{x}_{kh}; \mathbf{S}_k)$ denote the probability of $\mathbf{x}_{kh}$ with regard to the parameter $\mathbf{S}_k$, i.e., $P(\mathbf{x}_{kh}; \mathbf{S}_k) = \mathbf{S}_{k\mathbf{x}_{khs}\mathbf{x}_{kht}}$. Then, the log-likelihood function $L(\mathbf{S}_k; \mathcal{D}_k)$ is represented as follows:

$$L(\mathbf{S}_k; \mathcal{D}_k) = \log \prod_{h=1}^{n_k} P(\mathbf{x}_{kh}; \mathbf{S}_k) = \sum_{h=1}^{n_k} \log P(\mathbf{x}_{kh}; \mathbf{S}_k)$$

However, maximizing the classical likelihood $L(\mathbf{S}_k; \mathcal{D}_k)$ would be unsatisfactory since it is sensitive to noises or outliers in observations, and many networks such as knowledge graphs would be noisy and incomplete [127]. Considering this issue, we adopt maximum weighted likelihood estimation (MWLE) [128] that uses weights to vary the importance of $\log P(\mathbf{x}_{kh}; \mathbf{S}_k)$ for each observation. For the purpose, we define destination label weights $\{w_j | 1 \leq j \leq K\}$ that weigh the importance of $\log P(\mathbf{x}_{kh}; \mathbf{S}_k)$ according to destination label $\mathbf{x}_{kht}$. The intuition is that destination labels play a key role in relation inference since the relation between a source node and a destination node u is determined by the surfer's (destination) label when she arrives at u after starting from the source node. Thus, our weighted log-likelihood function is defined

as follows:

$$WL(\mathbf{S}_k; \mathcal{D}_k) = \sum_{h=1}^{n_k} w_{\mathbf{x}_{khl}} \log P(\mathbf{x}_{kh}; \mathbf{S}_k) \tag{5.1}$$

The following lemma provides the result of MWLE on the weighted log-likelihood function $WL(\mathbf{S}_k; \mathcal{D}_k)$.

Lemma 5.1 (Maximum Weighted Likelihood Estimation for Label Transition Probability Matrices)**.** *The following estimator* $\hat{\mathbf{S}}_{kij}$ *maximizes the weighted log-likelihood function* $WL(\mathbf{S}_k; \mathcal{D}_k)$ *in Equation* (5.1)*:*

$$\hat{\mathbf{S}}_{kij} = \frac{w_j N_{kij}}{\sum_{z=1}^{K} w_z N_{kiz}} \tag{5.2}$$

where $N_{kij} = \sum_{h=1}^{n_k} \mathbf{1}(\mathbf{x}_{khs} = i, \mathbf{x}_{kht} = j)$ *is the count for the label transition observations* $l_i \xrightarrow{l_k} l_j$*, and* $\mathbf{1}(\cdot)$ *returns* 1 *if a given predicate is true, or* 0 *otherwise.*

Proof. See the proof in Lemma 5.2 of Section 5.4.1. □

Lemma 5.1 indicates that $\hat{\mathbf{S}}_{kij}$ is determined by the label weight w_j and the count N_{kij} for the observations. Suppose all label weights $\{w_j\}$ are fixed to 1 (i.e., no label weights). Then $\mathbf{S}_{kij}$ depends only on the given observations. If we set w_j to a high value, the probabilities that the surfer's label changes to label l_j increase. We select proper $\{w_j\}$ that provide the best performance in the validation sets, as described in Section 5.5.2.

5.3.3 Multi-Labeled Random Walk with Restart

We describe our proposed model in the following definition:

Definition 5.3 (Multi-Labeled Random Walk with Restart)**.** *A labeled random surfer has label* l_i *among* K *edge labels at a node except at source node* s*. The surfer starts from*

source node s without any label. Suppose the surfer is currently at node u, and c is the restart probability with $0 < c < 1$. Then, the surfer performs one of the followings at each step:

- ***Multi-Labeled Random Walk.*** *The surfer randomly moves from node u to a neighboring node v with probability $1 - c$ through l_{uv}-labeled edge. If her label was l_i at node u, then her label changes to label l_j at node v according to label transition probability $P(l_i \xrightarrow{l_{uv}} l_j)$.*
- ***Restart.*** *The surfer restarts at source node s with probability c. The surfer becomes unlabeled.* ■

MuRWR measures K probabilities at each node. Let $\mathbf{R}_{ui}$ denote the probability that l_i-labeled surfer is at node u after MuRWR from s. If $\mathbf{R}_{ui}$ is higher than $\mathbf{R}_{uj}$ for $j \neq i$, this indicates that l_i-labeled surfer frequently visits node u, implying source node s is highly related by edge label l_i to node u. Thus $\mathbf{R}_{ui}$ is used for a relevance score between s and u for label l_i.

Our model MuRWR also considers multi-faceted paths between s and u, which is controlled by the restart probability c. If c is low, the labeled surfer visits u via paths of various lengths since the surfer prefers random walks to restart. On the other hand, if c is high, the surfer mainly visits u through relatively short paths due to frequent restarts. We will empirically study the effect of c on relation inference in the Experiment section (see Figure 5.6).

Note that when the unlabeled surfer moves from source node s to node u, her label becomes l_{su} since they are directly related with label l_{su} as depicted in Figure 5.2(a). To make notations consistent, we add a dummy label l_d indicating *unlabeled* at source node s. Then we set label transition probabilities $P(l_d \xrightarrow{l_{su}} l_{su})$ to 1 for each out-neighbor u of source node s (line 1 in Algorithm 11).

5.3.4 Formulation for MuRWR

We present formulations of MuRWR before proposing an algorithm for it. We derive a recursive equation for the probabilities $\mathbf{R}_{ui}$ (MuRWR scores).

Element-wise Representation. The MuRWR scores at a node are recursively determined by in-neighbors of the node. Figure 5.4 shows an example of the formulation for MuRWR probabilities where the figure depicts the part of a graph having two edge labels l_1 and l_2. In Figure 5.4, we mark an (edge label, probability) pair on each edge where the probability is for surfer's choosing the corresponding edge.

Let $\mathbf{R}^{(t)}_{u1}$ denote the probability that l_1-labeled surfer visits node u at time t after starting from source node s. In order that the surfer visits node u with label l_1 at time t, her label should be changed into l_1 when the surfer moves to node u from one of in-neighbors of u. For example, when she moves from node v to node u, her label changes to label l_1 with probability $\mathbf{R}^{(t-1)}_{v1}P(l_1 \xrightarrow{l_1} l_1) + \mathbf{R}^{(t-1)}_{v2}P(l_2 \xrightarrow{l_1} l_1)$. Considering the restart action with c, $\mathbf{R}^{(t)}_{u1}$ is determined as:

$$\mathbf{R}^{(t)}_{u1} = (1-c)\underbrace{\left[\frac{1}{2}\left(\mathbf{R}^{(t-1)}_{v1}P(l_1 \xrightarrow{l_1} l_1) + \mathbf{R}^{(t-1)}_{v2}P(l_2 \xrightarrow{l_1} l_1)\right) + \frac{1}{3}\left(\mathbf{R}^{(t-1)}_{w1}P(l_1 \xrightarrow{l_2} l_1) + \mathbf{R}^{(t-1)}_{w2}P(l_2 \xrightarrow{l_2} l_1)\right)\right]}_{\text{Labeled Random Walk}} + c\underbrace{\mathbf{1}(u = s, l_1 = l_d)}_{\text{Restart}}$$

where $\mathbf{1}(\cdot)$ returns 1 if a given predicate is true, or 0 otherwise. Note that $\mathbf{R}^{(t)}_{u2}$ is also determined similarly to the above equation. The general equation for $\mathbf{R}^{(t)}_{ui}$ is represented as:

$$\mathbf{R}^{(t)}_{ui} = (1-c)\sum_{v \in \overleftarrow{\mathbf{N}}_u}\left(\frac{1}{|\overrightarrow{\mathbf{N}}_v|}\sum_{j=1}^{K}\mathbf{R}^{(t-1)}_{vj}P(l_j \xrightarrow{l_{vu}} l_i)\right) + c\mathbf{1}(u = s, l_i = l_d) \tag{5.3}$$

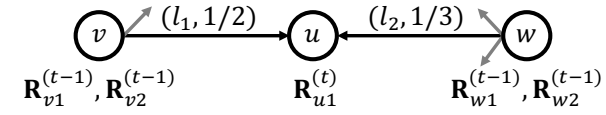


Figure 5.4: **Example of the formulation for the probability $\mathbf{R}_{u1}^{(t)}$** that l_1-labeled surfer visits node u at time t.

where $\overleftarrow{\mathbf{N}}_u$ is the set of in-neighbors of node u, and $\overrightarrow{\mathbf{N}}_v$ is the set of out-neighbors of node v. Note that $\mathbf{R}_{ui}^{(t)}$ is the accumulated result of MuRWR until step t with decaying factor $1-c$, as interpreted similarly in PageRank or RWR.

Matrix Representation. We represent Equation (5.3) in a matrix form using symbols in the following definitions:

Definition 5.4 (Labeled Semi-adjacency Matrix)**.** *The l_k-labeled semi-ad-jacency matrix $\mathbf{A}_k$ is a matrix such that (u,v)-th entry $\mathbf{A}_{kuv}$ of $\mathbf{A}_k$ is 1 if the label of the edge $u \to v$ is l_k, or 0 otherwise.* ■

Definition 5.5 (Labeled Semi-row-normalized Matrix)**.** *Let $\mathbf{D}$ be the out-degree diagonal matrix of a graph G where $\mathbf{D}_{uu}$ is the out-degree of node u. Then l_k-labeled semi-row-normalized matrix $\tilde{\mathbf{A}}_k$ is defined by $\tilde{\mathbf{A}}_k = \mathbf{D}^{-1}\mathbf{A}_k$. In other words, $\tilde{\mathbf{A}}_{kuv} = |\overrightarrow{\mathbf{N}}_u|^{-1} = \mathbf{D}_{uu}^{-1}$ if there is l_k-labeled edge from node u to node v in the graph G, or 0 otherwise, where $\overrightarrow{\mathbf{N}}_u$ is the set of out-neighbors of node u.* ■

Based on Definitions 5.2, 5.4 and 5.5, Equation (5.3) is rewritten as follows:

$$\mathbf{R}^{(t)} = (1-c)\underbrace{\sum_{k=1}^{K}(\tilde{\mathbf{A}}_k^{\top}\mathbf{R}^{(t-1)}\mathbf{S}_k)}_{\text{Labeled Random Walk}} + \underbrace{c\,\mathbf{Q}_s}_{\text{Restart}} \tag{5.4}$$

where $\mathbf{R}^{(t)} \in \mathbb{R}^{n\times K}$ is an MuRWR score matrix such that each entry $\mathbf{R}_{ui}^{(t)}$ is a score between source node s and node u for edge label l_i at step t, and $\mathbf{Q}_s \in \mathbb{R}^{n\times K}$ is a single entry matrix whose (s,d)-th entry is 1, and other entries are 0, where the index d is for the dummy label l_d. We provide the detailed derivation from Equation (5.3) to

Algorithm 9: Learning phase for MuRWR

Input: labeled adjacency matrix $\mathbf{A}$, and label weights $\{w_j\}$
Output: label transition probability matrices $\mathbf{S}_k$ for $1 \leq k \leq K$

1: enumerate all transitive triangles in the graph G represented by $\mathbf{A}$ using a triangle enumeration algorithm [126] to collect label transition observations $\mathcal{D}_k$ for $1 \leq k \leq K$
2: set $N_{kij} \leftarrow 0$ for $1 \leq i, j, k \leq K$
3: **for** $k = 1$ to K **do**
4: **for each** $\mathbf{x}_{kh} = (\mathbf{x}_{khs} \xrightarrow{l_k} \mathbf{x}_{kht}) \in \mathcal{D}_k$ **do**
5: set $i \leftarrow \mathbf{x}_{khs}$, $j \leftarrow \mathbf{x}_{kht}$, and $N_{kij} \leftarrow N_{kij} + 1$
6: **end for**
7: $\mathbf{S}_{kij} \leftarrow \frac{w_j N_{kij}}{\sum_{z=1}^{k} w_z N_{kiz}}$ for $1 \leq i, j \leq K$
8: **end for**
9: **return** $\mathbf{S}_k$ for $1 \leq k \leq K$

Equation (5.4) in Lemma 5.5 of Section 5.4.2.

Note that MuRWR is a generalized version of RWR, i.e., MuRWR works on edge-labeled graphs as well as plain graphs without edge labels, which is proved in the following.

Property 4. *MuRWR produces the same result with RWR in a plain graph.*

Proof. This case is equivalent to when the number K of edge labels is 1. Then, $\mathbf{R}^{(t)}$ and $\mathbf{Q}_s$ are written as vectors $\mathbf{r}^{(t)}$ and $\mathbf{q}_s$ in $\mathbb{R}^{n \times 1}$, respectively, where n is the number of nodes. Since $K = 1$, there is only one type of labeled transitive triangle; thus, $\mathbf{S}_1 \in \mathbb{R}^{1 \times 1} = 1$. Then, Equation (5.4) is exactly the same with $\mathbf{r}^{(t)} = (1-c)\tilde{\mathbf{A}}^\top \mathbf{r}^{(t-1)} + c\mathbf{q}_s$, the recursive equation of RWR [56]. □

5.3.5 Algorithm for MuRWR

Learning Phase (Algorithm 9). Given a labeled adjacency matrix $\mathbf{A}$, the learning phase learns the label transition probability matrices $\mathbf{S}_k$. We enumerate all transitive

Algorithm 10: Normalization phase for MuRWR

Input: labeled adjacency matrix $\mathbf{A}$

Output: labeled semi-row-normalized matrices $\tilde{\mathbf{A}}_k$ for $1 \leq k \leq K$

1: for each edge label l_k, construct semi-adjacency matrix $\mathbf{A}_k$ from the graph G represented by $\mathbf{A}$
2: compute adjacency matrix $\mathbf{A}' = \sum_k \mathbf{A}_k$ and out-degree diagonal matrix $\mathbf{D}$ such that $\mathbf{D}_{uu} = \sum_v \mathbf{A}'_{uv}$
3: for each edge label l_k, compute semi-row-normalized matrix $\tilde{\mathbf{A}}_k$, i.e., $\tilde{\mathbf{A}}_k = \mathbf{D}^{-1}\mathbf{A}_k$
4: **return** $\tilde{\mathbf{A}}_k$ for $1 \leq k \leq K$

Algorithm 11: Iterative Algorithm for MuRWR

Input: semi-row-normalized matrices $\tilde{\mathbf{A}}_k$, label transition probability matrices $\mathbf{S}_k$ for $1 \leq k \leq K$, source node s, restart probability c, and error tolerance ε

Output: MuRWR relevance score matrix $\mathbf{R}$

1: for each node $u \in \overrightarrow{\mathbf{N}}_s$, set $\mathbf{S}_{kij} \leftarrow 1$ where k and j are the indices for l_{su}, and i is for l_d, i.e., set $P(l_d \xrightarrow{l_{su}} l_{su}) = 1$
2: set starting matrix $\mathbf{Q}_s$ from source node s and $\mathbf{R}^{(0)} \leftarrow \mathbf{Q}_s$
3: set step $t \leftarrow 0$
4: **repeat**
5: update $t \leftarrow t+1$
6: compute $\mathbf{R}^{(t)} \leftarrow (1-c)\sum_{k=1}^{K}(\tilde{\mathbf{A}}_k^\top \mathbf{R}^{(t-1)}\mathbf{S}_k) + c\mathbf{Q}_s$
7: compute residual $\delta^{(t)} = \|\mathbf{R}^{(t)} - \mathbf{R}^{(t-1)}\|_{1,1}$
8: **until** $\delta^{(t)} < \varepsilon$
9: **return** MuRWR relevance score matrix $\mathbf{R} \leftarrow \mathbf{R}^{(t)}$

triangles from the graph represented by $\mathbf{A}$ (line 1) using a triangle enumeration algorithm [126]. Based on the enumerated transitive triangles, we estimate the label transition matrices $\mathbf{S}_k$ using Equation (5.2) (lines 2∼8).

Normalization Phase (Algorithm 10). This phase produces the semi-row normalized matrices $\tilde{\mathbf{A}}_k$ for $1 \leq k \leq K$ from the labeled adjacency matrix $\mathbf{A}$ after building semi-adjacency matrices $\mathbf{A}_k$ by Definitions 5.4 and 5.5 (lines 1∼3).

Iteration Phase (Algorithm 11). This phase computes the MuRWR relevance score matrix $\mathbf{R}$ w.r.t. source node s. We first set $P(l_d \xrightarrow{l_{su}} l_{su}) = 1$ for each $u \in \overrightarrow{\mathbf{N}}_s$ for the dummy label l_d (line 1). After setting starting matrix $\mathbf{Q}_s$ and initializing $\mathbf{R}^{(0)}$ to

$\mathbf{Q}_s$ (line 2), we repeat the update for $\mathbf{R}^{(t)}$ based on Equation (5.4) until convergence (lines 4∼8). We compute residual $\delta^{(t)}$ between $\mathbf{R}^{(t)}$ and $\mathbf{R}^{(t-1)}$ which is the result from the previous iteration (line 7) where $\delta^{(t)}$ is measured by entry-wise L1 matrix norm, i.e., $\|\mathbf{A}\|_{1,1} = \sum_{i,j} |\mathbf{A}_{ij}|$. This stops when $\delta^{(t)}$ is smaller than error tolerance ε.

Convergence Analysis. We prove the convergence guarantee of the iterative method (Algorithm 11) of MuRWR in Theorem 5.1.

Theorem 5.1 (Convergence of MuRWR). *Suppose* $\mathbf{r}^{(t)} = vec(\mathbf{R}^{(t)})$ *and* $\mathbf{q}_s = vec(\mathbf{Q}_s)$ *where* $\mathbf{R}^{(t)}$ *is the MuRWR score matrix at step* t *in Algorithm 11, and* $vec(\cdot)$ *is the vec-operator which converts a matrix into a vector [129]. Then the residual* $\delta^{(t)} \leq 2(1-c)^t$, *and* $\mathbf{r}^{(t)}$ *converges to* $\mathbf{r} = c(\mathbf{I} - (1-c)\tilde{\mathbf{B}}^\top)^{-1}\mathbf{q}_s$ *where* $\tilde{\mathbf{B}}^\top = \sum_{k=1}^{K} \mathbf{S}_k^\top \otimes \tilde{\mathbf{A}}_k^\top$ *and* $\otimes$ *is Kronecker product.*

Proof. Equation (5.4) is vectorized by $vec(\cdot)$ as follows:

$$\begin{aligned} vec(\mathbf{R}^{(t)}) &= (1-c)\sum_{k=1}^{K} vec(\tilde{\mathbf{A}}_k^\top \mathbf{R}^{(t-1)} \mathbf{S}_k) + c(vec(\mathbf{Q}_s)) \\ &= (1-c)\sum_{k=1}^{K} (\mathbf{S}_k^\top \otimes \tilde{\mathbf{A}}_k^\top) vec(\mathbf{R}^{(t-1)}) + c(vec(\mathbf{Q}_s)) \\ \Leftrightarrow \mathbf{r}^{(t)} &= (1-c)\tilde{\mathbf{B}}^\top \mathbf{r}^{(t-1)} + c\mathbf{q}_s \end{aligned} \tag{5.5}$$

where $\tilde{\mathbf{B}}^\top = \sum_{k=1}^{K} \mathbf{S}_k^\top \otimes \tilde{\mathbf{A}}_k^\top$. The second equation is derived by $vec(\mathbf{ABC}) = (\mathbf{C}^\top \otimes \mathbf{A})vec(\mathbf{B})$ [129].

Residual analysis. The residual $\delta^{(t)} = \|\mathbf{R}^{(t)} - \mathbf{R}^{(t-1)}\|_{1,1}$ is equal to $\|\mathbf{r}^{(t)} - \mathbf{r}^{(t-1)}\|_1$

since $\|\mathbf{A}\|_{1,1} = \|vec(\mathbf{A})\|_1$ [129]. Thus, the residual is bounded as follows:

$$\begin{aligned}
\delta^{(t)} &= \|\mathbf{r}^{(t)} - \mathbf{r}^{(t-1)}\|_1 = \|(1-c)\tilde{\mathbf{B}}^\top \mathbf{r}^{(t-1)} - (1-c)\tilde{\mathbf{B}}^\top \mathbf{r}^{(t-2)}\|_1 \\
&\leq (1-c)\|\tilde{\mathbf{B}}^\top\|_1 \|\mathbf{r}^{(t-1)} - \mathbf{r}^{(t-2)}\|_1 \\
&\leq (1-c)\|\mathbf{r}^{(t-1)} - \mathbf{r}^{(t-2)}\|_1 \leq \cdots \\
&\leq (1-c)^{t-1}\|\mathbf{r}^{(1)} - \mathbf{r}^{(0)}\|_1 = (1-c)^t \|\tilde{\mathbf{B}}^\top \mathbf{q}_s - \mathbf{q}_s\|_1 \leq 2(1-c)^t
\end{aligned}$$

Note that $\|\tilde{\mathbf{B}}^\top \mathbf{q}_s - \mathbf{q}_s\|_1 \leq 2$ since $\|\tilde{\mathbf{B}}^\top \mathbf{q}_s - \mathbf{q}_s\|_1 \leq \|\tilde{\mathbf{B}}^\top \mathbf{q}_s\|_1 + \|\mathbf{q}_s\|_1 \leq \|\tilde{\mathbf{B}}^\top\|_1 \|\mathbf{q}_s\|_1 + 1 \leq 2$ where $\|\mathbf{q}_s\|_1 = 1$, and $\|\tilde{\mathbf{B}}^\top\|_1 \leq 1$ by Lemma 5.6. Hence, the bound for the residual is $\delta^{(t)} \leq 2(1-c)^t$.

Convergence analysis. Equation (5.5) is also represented as follows:

$$\begin{aligned}
\mathbf{r}^{(t)} &= (1-c)\tilde{\mathbf{B}}^\top \mathbf{r}^{(t-1)} + c\mathbf{q}_s \\
&= \left((1-c)\tilde{\mathbf{B}}^\top\right)^2 \mathbf{r}^{(t-2)} + c\left((1-c)\tilde{\mathbf{B}}^\top + \mathbf{I}\right)\mathbf{q}_s = \cdots \\
&= \left((1-c)\tilde{\mathbf{B}}^\top\right)^t \mathbf{r}^{(0)} + c\sum_{j=0}^{t-1}\left((1-c)\tilde{\mathbf{B}}^\top\right)^j \mathbf{q}_s
\end{aligned}$$

The spectral radius $\rho((1-c)\tilde{\mathbf{B}}^\top) \leq (1-c) < 1$ when $0 < c < 1$ since $\rho(\tilde{\mathbf{B}}^\top) \leq 1$ according to Lemma 5.6 of Section 5.4.3. Hence, $\lim_{t\to\infty}((1-c)\tilde{\mathbf{B}}^\top)^t \mathbf{r}^{(0)} = \mathbf{0}$ [115] and $\lim_{t\to\infty} \mathbf{r}^{(k)}$ converges as:

$$\lim_{t\to\infty} \mathbf{r}^{(t)} = c\sum_{j=0}^{\infty}\left((1-c)\tilde{\mathbf{B}}^\top\right)^j \mathbf{q}_s = c(\mathbf{I} - (1-c)\tilde{\mathbf{B}}^\top)^{-1}\mathbf{q}_s$$

$\sum_{j=0}^{\infty}((1-c)\tilde{\mathbf{B}}^\top)^j$ is a geometric series of the matrix $(1-c)\tilde{\mathbf{B}}^\top$, and it converges to $(\mathbf{I} - (1-c)\tilde{\mathbf{B}}^\top)^{-1}$ since the spectral radius of $(1-c)\tilde{\mathbf{B}}^\top$ is less than 1 [115]. Note that $\mathbf{I} - (1-c)\tilde{\mathbf{B}}^\top$ is invertible if $\rho((1-c)\tilde{\mathbf{B}}^\top) \leq (1-c) < 1$ [115]. □

Theorem 5.1 implies that as step t increases, the residual $\delta^{(t)}$ in Algorithm 11 monotonically decreases, and converges to zero since $0 < c < 1$ as specified in Definition 5.3. Also, $\mathbf{r}^{(t)}$ converges to a unique solution $\mathbf{r} = c(\mathbf{I} - (1-c)\tilde{\mathbf{B}}^{\top})^{-1}\mathbf{q}_s$. We analyze the time complexities of the algorithms of MuRWR in Section 5.4.4.

5.4 Theoretical Results

We provides the lemma and proofs used for modeling MuRWR and analyzing the algorithms for MuRWR.

5.4.1 Lemma for Solution of Label Transition Probabilities and Convexity

Lemma 5.2. *The estimator in Equation* (5.2) *maximizes the weighted log-likelihood function* $WL(\mathbf{S}_k; \mathcal{D}_k)$ *in Equation* (5.1).

Proof. Our goal is to find $\mathbf{S}_k$ that maximizes the weighted log-likelihood function $WL(\mathbf{S}_k; \mathcal{D}_k)$, which is equivalent to minimizing $-WL(\mathbf{S}_k; \mathcal{D}_k)$. We first rewrite the probability $P(\mathbf{x}_{kh}; \mathbf{S}_k)$ as follows:

$$P(\mathbf{x}_{kh}; \mathbf{S}_k) = \mathbf{S}_{k\mathbf{x}_{khs}\mathbf{x}_{kht}} = \prod_{i=1}^{K}\prod_{j=1}^{K}(\mathbf{S}_{kij})^{\mathbf{1}(\mathbf{x}_{khs}=i, \mathbf{x}_{kht}=j)}$$

Note that $\mathbf{x}_{kh}$ is a label transition observation $\mathbf{x}_{khs} \xrightarrow{l_k} \mathbf{x}_{kht}$ such that $\mathbf{x}_{khs}, \mathbf{x}_{kht} \in \mathbf{L}$ where $\mathbf{x}_{khs}$ is the surfer's source label, and $\mathbf{x}_{kht}$ is her destination label. Then, $-WL(\mathbf{S}_k; \mathcal{D}_k)$

is represented as follows:

$$
\begin{aligned}
-WL(\mathbf{S}_k;\mathcal{D}_k) &= -\sum_{h=1}^{n_k} w_{\mathbf{x}_{khl}} \log P(\mathbf{x}_{kh};\mathbf{S}_k) \\
&= -\sum_{h=1}^{n_k}\sum_{i=1}^{K}\sum_{j=1}^{K} w_j \mathbf{1}(\mathbf{x}_{khs}=i,\mathbf{x}_{kht}=j)\log \mathbf{S}_{kij} \\
&= -\sum_{i=1}^{K}\sum_{j=1}^{K} w_j \left(\sum_{h=1}^{n_k} \mathbf{1}(\mathbf{x}_{khs}=i,\mathbf{x}_{kht}=j)\right)\log \mathbf{S}_{kij} \\
&= -\sum_{i=1}^{K}\sum_{j=1}^{K} w_j N_{kij}\log \mathbf{S}_{kij}
\end{aligned}
$$

where $N_{kij} = \sum_{h=1}^{n_k} \mathbf{1}(\mathbf{x}_{khs}=i,\mathbf{x}_{kht}=j)$ is the count of the label transition observations. Then, the minimization problem is represented as follows:

$$
\begin{aligned}
\underset{\mathbf{S}_{kij}}{\text{minimize}} \quad & -WL(\mathbf{S}_k;\mathcal{D}_k) = -\sum_{i=1}^{K}\sum_{j=1}^{K} w_j N_{kij}\log \mathbf{S}_{kij} \\
\text{subject to} \quad & \mathbf{S}_{kij} \geq 0 \text{ for } 1 \leq i,j \leq K, \\
& \sum_{j=1}^{K} \mathbf{S}_{kij} = 1 \text{ for } 1 \leq i \leq K.
\end{aligned}
\tag{5.6}
$$

Note that the above problem is convex (see Lemma 5.3); thus, the optimization problem is solved by the KKT theorem [99], and the solution of the problem is represented as Equation (5.2) (details in Lemma 5.4). □

Lemma 5.3. *The optimization problem in Equation* (5.6) *is convex.*

Proof. The objective function is convex since the negative log functions $-\log \mathbf{S}_{kij}$ are convex, and the sum of non-negatively weighted convex functions is convex (i.e., $w_j N_{kij} \geq 0$) [99]. Let $\mathbf{C}$ be a set of $\mathbf{S}_k$ satisfying the constraints, i.e., $\mathbf{C} = \{\mathbf{S}_k | \mathbf{S}_k\mathbf{1} = \mathbf{1}, \mathbf{S}_{kij} \geq 0, \text{ for } 1 \leq i,j \leq K\}$. For $\mathbf{S}_{k_1}, \mathbf{S}_{k_2} \in \mathbf{C}$ and $\theta_1 + \theta_2 = 1$ such that $\theta_1, \theta_2 \geq 0$, let

$\mathbf{S}_{k_3} = \theta_1 \mathbf{S}_{k_1} + \theta_2 \mathbf{S}_{k_2}$. Then $\mathbf{S}_{k_3}\mathbf{1} = (\theta_1 \mathbf{S}_{k_1} + \theta_2 \mathbf{S}_{k_2})\mathbf{1} = \mathbf{1}$ indicating $\mathbf{S}_{k_3} \in \mathbf{C}$. Thus $\mathbf{C}$ is convex by the definition of convex set [99]. □

Lemma 5.4. *The solution of the optimization problem in Equation* (5.6) *is represented as Equation* (5.2).

Proof. The lagrangian $\mathcal{L}(\cdot)$ of the objective function in Equation (5.6) is represented as follows:

$$\mathcal{L}(\mathbf{S}_k, \lambda, \nu) = -\sum_{i=1}^{K}\sum_{j=1}^{K} w_j N_{kij} \log \mathbf{S}_{kij} + \sum_{i=1}^{K}\sum_{j=1}^{K} -\lambda_{ij}\mathbf{S}_{kij} + \sum_{i=1}^{K} \nu_i \sum_{j=1}^{K} \left(\mathbf{S}_{kij} - 1\right)$$

where λ and ν are inequality and equality lagrange multipliers, respectively. Let $\hat{\mathbf{S}}_{kij}$ be the solution that minimizes Equation (5.6). λ^* and ν^* denote the optional points for λ and ν, respectively. The stationarity condition $\nabla_{\mathbf{S}_k}\mathcal{L}(\hat{\mathbf{S}}_k, \lambda^*, \nu^*) = 0$ implies the following equation:

$$\frac{\partial \mathcal{L}(\hat{\mathbf{S}}_k, \lambda^*, \nu^*)}{\partial \mathbf{S}_{kij}} = -\frac{w_j N_{kij}}{\hat{\mathbf{S}}_{kij}} - \lambda^*_{ij} + \nu^*_i = 0 \Leftrightarrow \hat{\mathbf{S}}_{kij} = \frac{w_j N_{kij}}{\nu^*_i - \lambda^*_{ij}}$$

By the complementary slackness $\lambda^*_{ij}\hat{\mathbf{S}}_{kij} = 0$, primal feasibility $\hat{\mathbf{S}}_{kij} \geq 0$, and dual feasibility $\lambda^*_{ij} \geq 0$,

- $\hat{\mathbf{S}}_{kij} > 0 \Rightarrow \lambda^*_{ij} = 0 \Leftrightarrow \hat{\mathbf{S}}_{kij} = \dfrac{w_j N_{kij}}{\nu^*_i} > 0 \Leftrightarrow w_j N_{kij} \neq 0$
- $\lambda^*_{ij} > 0 \Rightarrow \hat{\mathbf{S}}_{kij} = 0 \Leftrightarrow \hat{\mathbf{S}}_{kij} = \dfrac{w_j N_{kij}}{\nu^*_i - \lambda^*_{ij}} = 0 \Leftrightarrow w_j N_{kij} = 0$

For the case that $\hat{\mathbf{S}}_{kij} > 0$, ν^*_i is obtained from the equality constraint $\sum_{z=1}^{K} \hat{\mathbf{S}}_{kiz} = 1$

as follows:

$$\sum_{z=1}^{K} \hat{\mathbf{S}}_{kiz} = \sum_{\{z|\hat{\mathbf{S}}_{kiz}>0\}} \hat{\mathbf{S}}_{kiz} = \sum_{\{z|\hat{\mathbf{S}}_{kiz}>0\}} \frac{w_z N_{kiz}}{\nu_i^*} = 1 \Leftrightarrow$$

$$\nu_i^* = \sum_{\{z|\hat{\mathbf{S}}_{kiz}>0\}} w_z N_{kiz} = \sum_{\{z|\hat{\mathbf{S}}_{kiz}>0\}} w_z N_{kiz} + \sum_{\{z|\hat{\mathbf{S}}_{kiz}=0\}} w_z N_{kiz} = \sum_{z=1}^{K} w_z N_{kiz}$$

Hence, $\hat{\mathbf{S}}_{kij} = w_j N_{kij} / \nu_i^* = w_j N_{kij} / (\sum_{z=1}^{K} w_z N_{kiz})$ □

5.4.2 Lemma for Recursive Equation of MuRWR Score Matrix

Lemma 5.5. *Equation* (5.3) *is represented as Equation* (5.4).

Proof. In Equation (5.3), let l_p denote l_{vu}. For edge $v \to u$,

$$\sum_{j=1}^{K} \mathbf{R}_{vj} P(l_j \xrightarrow{l_{vu}} l_i) = \sum_{j=1}^{K} \mathbf{R}_{vj} P(l_j \xrightarrow{l_p} l_i) = \sum_{j=1}^{K} \mathbf{R}_{vj} \mathbf{S}_{pji}$$

where $\mathbf{S}_{pji}$ is the label transition probability $P(l_j \xrightarrow{l_p} l_i)$. By Definition 5.5, $\tilde{\mathbf{A}}_{pvu} = |\overrightarrow{\mathbf{N}}_v|^{-1} = \tilde{\mathbf{A}}_{puv}^{\top}$ for all p when $\tilde{\mathbf{A}}_{puv}$ is non-zero. Hence,

$$\sum_{v \in \overleftarrow{\mathbf{N}}_u} \left(\frac{1}{|\overrightarrow{\mathbf{N}}_v|} \sum_{j=1}^{K} \mathbf{R}_{vj} \mathbf{S}_{pji} \right) = \sum_{v \in \overleftarrow{\mathbf{N}}_u} \tilde{\mathbf{A}}_{puv}^{\top} \sum_{j=1}^{K} \mathbf{R}_{vj} \mathbf{S}_{pji} \tag{5.7}$$

Let $\overleftarrow{\mathbf{N}}_u^{(i)}$ be the set of in-neighbors of node u such that node $v \in \overleftarrow{\mathbf{N}}_u^{(i)}$ is connected to node u with edge label l_i. Then, $\overleftarrow{\mathbf{N}}_u$ is represented as $\overleftarrow{\mathbf{N}}_u = \overleftarrow{\mathbf{N}}_u^{(1)} \cup \cdots \cup \overleftarrow{\mathbf{N}}_u^{(K)}$. If there is no l_i-labeled edge to node u from any in-neighbor node, then $\overleftarrow{\mathbf{N}}_u^{(i)}$ is empty, i.e., $\overleftarrow{\mathbf{N}}_u^{(i)} = \emptyset$. Note that if node u is connected from node $v \in \overleftarrow{\mathbf{N}}_u^{(1)}$, then edge label l_{vu}

is l_1, i.e., in this case, $l_{vu} = l_p = l_1$. Thus, Equation (5.7) is represented as follows:

$$\begin{aligned}\sum_{v \in \overleftarrow{\mathbf{N}}_u} \tilde{\mathbf{A}}^{\top}_{puv} \sum_{j=1}^{K} \mathbf{R}_{vj} \mathbf{S}_{pji} &= \sum_{v \in \overleftarrow{\mathbf{N}}_u^{(1)}} \tilde{\mathbf{A}}^{\top}_{1uv} \sum_{j=1}^{K} \mathbf{R}_{vj} \mathbf{S}_{1ji} + \cdots + \sum_{v \in \overleftarrow{\mathbf{N}}_u^{(K)}} \tilde{\mathbf{A}}^{\top}_{Kuv} \sum_{j=1}^{K} \mathbf{R}_{vj} \mathbf{S}_{Kji} \\ &= \sum_{k=1}^{K} \left(\sum_{v \in \overleftarrow{\mathbf{N}}_u^{(k)}} \tilde{\mathbf{A}}^{\top}_{kuv} \sum_{j=1}^{K} \mathbf{R}_{vj} \mathbf{S}_{kji} \right) \end{aligned} \tag{5.8}$$

Let $(\cdot)_{ij}$ be (i,j)-th entry of a matrix. Then, $\sum_{v \in \overleftarrow{\mathbf{N}}_u^{(k)}} \tilde{\mathbf{A}}^{\top}_{kuv} \sum_{j=1}^{K} \mathbf{R}_{vj} \mathbf{S}_{kji}$ in the above equation is written as:

$$\sum_{v \in \overleftarrow{\mathbf{N}}_u^{(k)}} \tilde{\mathbf{A}}^{\top}_{kuv} \sum_{j=1}^{K} \mathbf{R}_{vj} \mathbf{S}_{kji} = \sum_{v \in \overleftarrow{\mathbf{N}}_u^{(k)}} \tilde{\mathbf{A}}^{\top}_{kuv} (\mathbf{R}\mathbf{S}_k)_{vi} = (\tilde{\mathbf{A}}^{\top}_k \mathbf{R}\mathbf{S}_k)_{ui}$$

Then, Equation (5.8) is represented as follows:

$$\sum_{k=1}^{K} \left(\sum_{v \in \overleftarrow{\mathbf{N}}_u^{(k)}} \tilde{\mathbf{A}}^{\top}_{kuv} \sum_{j=1}^{K} \mathbf{R}_{vj} \mathbf{S}_{kji} \right) = \sum_{k=1}^{K} (\tilde{\mathbf{A}}^{\top}_k \mathbf{R}\mathbf{S}_k)_{ui} = \left(\sum_{k=1}^{K} \tilde{\mathbf{A}}^{\top}_k \mathbf{R}\mathbf{S}_k \right)_{ui}$$

Thus, $\mathbf{R}_{ui}$ in Equation (5.3) is written as follows:

$$\mathbf{R}_{ui} = (1-c) \left(\sum_{k=1}^{K} \tilde{\mathbf{A}}^{\top}_k \mathbf{R}\mathbf{S}_k \right)_{ui} + c\mathbf{1}(u = s, l_i = l_d)$$

For $1 \leq u \leq n$ and $1 \leq i \leq K$ where n is the number of nodes, the above equation is represented as Equation (5.4). □

5.4.3 Lemma for Spectral Radius in Convergence Theorem

Lemma 5.6. *Suppose* $\tilde{\mathbf{B}}^\top = \sum_{k=1}^{K} \mathbf{S}_k^\top \otimes \tilde{\mathbf{A}}_k^\top$ *where* $\tilde{\mathbf{A}}_k$ *is* k*-th labeled semi-row-normalized matrix, and* $\mathbf{S}_k$ *is* k*-th label transition probability matrix. Then,* $\|\tilde{\mathbf{B}}^\top\|_1 \leq 1$*, and the spectral radius of* $\tilde{\mathbf{B}}^\top$ *is bounded as follows:* $\rho(\tilde{\mathbf{B}}^\top) \leq 1$.

Proof. According to spectral radius theorem [78], $\rho(\tilde{\mathbf{B}}^\top) \leq \|\tilde{\mathbf{B}}^\top\|_1$ where $\|\tilde{\mathbf{B}}^\top\|_1$ is the maximum absolute column sum of $\tilde{\mathbf{B}}^\top$. Since each entry of $\tilde{\mathbf{B}}^\top$ is non-negative, $\|\tilde{\mathbf{B}}^\top\|_1$ is equal to the maximum value of the column sums of the matrix. The column sums are represented as follows:

$$
\begin{aligned}
(\mathbf{1}^\top \otimes \mathbf{1}^\top)\tilde{\mathbf{B}}^\top &= (\mathbf{1}^\top \otimes \mathbf{1}^\top)(\sum_{k=1}^{K} \mathbf{S}_k^\top \otimes \tilde{\mathbf{A}}_k^\top) \\
&= \sum_{k=1}^{K} (\mathbf{1}^\top \otimes \mathbf{1}^\top)(\mathbf{S}_k^\top \otimes \tilde{\mathbf{A}}_k^\top) = \sum_{k=1}^{K} \mathbf{1}^\top \mathbf{S}_k^\top \otimes \mathbf{1}^\top \tilde{\mathbf{A}}_k^\top
\end{aligned}
$$

According to Definition 5.2, the sum of each row of $\mathbf{S}_k$ is 1, i.e., $\mathbf{S}_k \mathbf{1} = \mathbf{1} \Leftrightarrow \mathbf{1}^\top \mathbf{S}_k^\top = \mathbf{1}^\top$. Hence,

$$
\sum_{k=1}^{K} \mathbf{1}^\top \mathbf{S}_k^\top \otimes \mathbf{1}^\top \tilde{\mathbf{A}}_k^\top = \sum_{k=1}^{K} \mathbf{1}^\top \otimes \mathbf{1}^\top \tilde{\mathbf{A}}_k^\top = \mathbf{1}^\top \otimes \sum_{k=1}^{K} \mathbf{1}^\top \tilde{\mathbf{A}}_k^\top = \mathbf{1}^\top \otimes \mathbf{1}^\top \sum_{k=1}^{K} \tilde{\mathbf{A}}_k^\top
$$

Note that $\tilde{\mathbf{A}}_k^\top = \mathbf{A}_k^\top \mathbf{D}^{-1}$ according to Definition 5.4. Suppose $\mathbf{A}' = \sum_{k=1}^{K} \mathbf{A}_k$ is the adjacency matrix of the graph G without edge labels. Then, the following holds: $\mathbf{1}^\top \sum_{k=1}^{K} \tilde{\mathbf{A}}_k^\top = \mathbf{1}^\top \sum_{k=1}^{K} \mathbf{A}_k^\top \mathbf{D}^{-1} = (\mathbf{A}'\mathbf{1})^\top \mathbf{D}^{-1}$. The u-th row of $\mathbf{A}'\mathbf{1}$ indicates the out-degree of node u, denoted by deg_u. If node u is a deadend node, then $deg_u = 0$. Otherwise, $deg_u > 0$. Note that $\mathbf{D}$ is the out-degree diagonal matrix of G and $\mathbf{D}_{uu}^{-1} = 1/deg_u$ if node u is not a deadend. Otherwise, $\mathbf{D}_{uu}^{-1} = 0$. Thus, $(\mathbf{A}'\mathbf{1})^\top \mathbf{D}^{-1} = \mathbf{b}^\top$ where u-th entry of $\mathbf{b}$ is 1 if node u is non-deadend, or 0 otherwise. Hence, $(\mathbf{1}^\top \otimes \mathbf{1}^\top)\tilde{\mathbf{B}}^\top = \mathbf{1}^\top \otimes \mathbf{b}^\top$ which is the column sum vector of $\tilde{\mathbf{B}}^\top$, and the maximum value of the vector is less

than or equal to 1. Therefore, $\|\tilde{\mathbf{B}}^\top\|_1 \leq 1$, implying $\rho(\tilde{\mathbf{B}}^\top) \leq \|\tilde{\mathbf{B}}^\top\|_1 \leq 1$. □

5.4.4 Lemma for Complexity Analysis

Lemma 5.7. *The time complexity of Algorithms 9 and 10 is $O(m^{1.5} + K^3)$ where m is the number of edges, and K is the number of edge labels.*

Proof. In Algorithm 1, it takes $O(m^{1.5})$ time to enumerate all transitive triangles in the given graph G using a triangle enumeration algorithm [126] (line 1 in Algorithm 9). Also, estimating $\mathbf{S}_k$ requires $O(K^3)$ time (lines 2 $\sim$ 8 in Algorithm 9). Algorithm 10 takes $O(m)$ time for counting out-degrees of nodes (line 2 in Algorithm 10) and computing $\mathbf{A}_{-k} = \mathbf{D}^{-1}\mathbf{A}_k$ for $1 \leq k \leq K$ (line 3 in Algorithm 10). □

Lemma 5.8. *The time complexity of Algorithm 11 is $O(T(Km + K^3 n))$ where $T = \log_{(1-c)} \frac{\varepsilon}{2}$ indicates the number of iterations for convergence, ε is an error tolerance, m is the number of edges, n is the number of nodes, and K is the number of edge labels.*

Proof. Let m_k denote the number of non-zeros in k-th semi-row normalized matrix $\mathbf{A}_{-k}$ stored in a sparse matrix format such as compressed column storage (CCS). For each iteration, it takes $O(Km_k + K^2 n)$ time to compute $\tilde{\mathbf{A}}_k^\top \mathbf{R}^{(t-1)} \mathbf{S}_k$ since the sparse matrix product $\tilde{\mathbf{A}}_k^\top \mathbf{R}^{(t-1)}$ requires $O(Km_k)$ time, and the dense matrix product $(\tilde{\mathbf{A}}_k^\top \mathbf{R}^{(t-1)})\mathbf{S}_k$ takes $O(K^2 n)$ time. Thus, it takes $O(\sum_{k=1}^{K}(Km_k + K^2 n)) = O(Km + K^3 n)$ time to compute $\sum_{k=1}^{K}(\tilde{\mathbf{A}}_k^\top \mathbf{R}^{(t-1)} \mathbf{S}_k)$ where $\sum_{k=1}^{K} m_k = m$ (line 6). Note that when $2(1-c)^t \leq \varepsilon$, $\mathbf{R}^{(t)}$ converges since $\delta^{(t)} \leq 2(1-c)^t$ by Theorem 5.1. Hence, for $t \geq \log_{(1-c)} \frac{\varepsilon}{2}$, the iteration is necessarily terminated. Thus, the number of iterations for convergence is estimated at $\log_{(1-c)} \frac{\varepsilon}{2}$, and the total time complexity of Algorithm 11 is $O((\log_{(1-c)} \frac{\varepsilon}{2})(Km + K^3 n))$. □

Table 5.2: **Statistics of the datasets used in Chapter 5** where n is the number of nodes, m is the number of edges, and K is the number of edge labels.

Dataset	n	m	K	Description
Epinions[1]	131,828	841,372	2	Signed social network
Slashdot[2]	79,120	515,397	2	Signed social network
WN18[3]	40,943	151,433	18	Knowledge graph
WN11[4]	38,588	138,887	11	Knowledge graph
WikiVote[5]	7,118	103,675	2	Signed voting network
Advogato[6]	6,541	47,135	3	Social network

[1] http://www.trustlet.org/wiki/Extended_Epinions_dataset
[2] http://dai-labor.de/IRML/datasets
[3] https://everest.hds.utc.fr/doku.php?id=en:transe
[4] http://cs.stanford.edu/ danqi/data/nips13-dataset.tar.bz2
[5] http://snap.stanford.edu/data/wiki-Vote.html
[6] http://konect.uni-koblenz.de/networks/advogato

5.5 Experiment

We perform experiments to answer the following questions:

- **Q1. Performance of Relation Inference (Section 5.5.2).** How accurately does MuRWR predict edge labels between nodes compared to other existing methods?
- **Q2. Effects of Label Weights (Section 5.5.3).** How does the label weights w_j in MuRWR affect the predictive performance of MuRWR for the relation inference task?
- **Q3. Effects of Restart Probability (Section 5.5.4).** How does the restart probability c in MuRWR affect the inference performance of MuRWR?
- **Q4. Convergence (Section 5.5.5).** Does the iterative algorithm for MuRWR converge? How does the restart probability c affect the convergence behavior of the algorithm?

5.5.1 Experimental Settings

Datasets. The datasets used for our experiments are summarized in Table 5.2. In the Epinions [130] and the Slashdot [63] datasets, users rate each other positively or negatively. The WN11 [131] and WN18 [132] datasets are from WordNet [122], a knowledge graph of words where an edge label is a relation between words. The WikiVote dataset is a signed network where users vote positively or negatively on their candidates [133]. The Advogato dataset is a social network where an edge label indicates a level of trust [134].

Competitors. We compare our model MuRWR to the following methods:

- **Random**: Random predicts the label of a test edge randomly, which provides the worst performance of each dataset.
- **LINE** [135] and **node2vec** [136]: We exploit well-known embedding models LINE and node2vec as baseline methods, although they are not designed for relation inference. Since they are designed for plain networks, we first extract an embedding vector of each node using those methods in a given network without edge labels. Next, we convert l_i-edge between two nodes into a training instance, i.e., the feature is the concatenation of the embeddings of those nodes, and the label is l_i. We perform multinomial logistic regression based on a softmax function to predict the edge label.
- **MRWR** [65] and **SRWR** [22]: We compare MuRWR to MRWR and SRWR which are RWR based variants. Although MRWR is originally designed for signed networks, it is easy to make the method work on edge-labeled graphs by computing RWR on each subgraph containing only a specific edge label. Note that SRWR cannot work on general edge-labeled graphs since it is designed

only for signed networks (i.e., $K = 2$); hence, the results of SRWR for other edge-labeled graphs (i.e., $K > 2$) are omitted in Table 5.3 and 5.4.

- **PRA** [67]: PRA is a path feature model used for relation inference in edge-labeled graphs. PRA extracts path features between two nodes using a random surfer, and exploits those features with logistic regressors.
- **TransE** [92] and **TransR** [93]: TransE is a translation based method which considers the relation l between nodes s and t as a translation between the corresponding node embeddings. Specifically, TransE discovers embeddings $\mathbf{s}$, $\mathbf{l}$, and $\mathbf{t}$ minimizing $f(s,l,t) = \|\mathbf{s}+\mathbf{l}-\mathbf{t}\|_2$. Given nodes s and t, it predicts their relation l that minimizes $f(s,l,t)$. TransR models entities and relations in distinct spaces, i.e., $\mathbf{s}_l = \mathbf{s}\mathbf{M}_l$ and $\mathbf{t}_l = \mathbf{t}\mathbf{M}_l$ where $\mathbf{M}_l$ is projection matrix for l. Then, TransR minimizes $f(s,l,t) = \|\mathbf{s}_l+\mathbf{l}-\mathbf{t}_l\|_2$.

5.5.2 Relation Inference Task

We evaluate our proposed model MuRWR on a relation inference task defined as follows: given an edge-labeled graph containing missed labels of edges, predict those edge labels. We randomly select 500 source nodes and choose 20% of out-going edges from each source node as a validation set which is used for selecting proper hyper-parameters of each method. For a test set, we randomly select another 500 source nodes and choose 20% of out-going edges from each source node. Then we remove each selected edge $s \to t$, and predict the edge's label using relevance scores w.r.t. source node s, i.e., the predicted label $\hat{l}$ is decided as follows: $\hat{l} = \text{argmax}_{1 \le k \le K} \mathbf{R}_{tk}$. We measure the prediction accuracy for the test dataset, which is defined as follows: *accuracy* $=$ *# of correct predictions*/*# of test edges*. Also this task is considered as multi-class classification (i.e., classify the label of a test edge); thus, we measure

Table 5.3: **Performance of relation inference in terms of accuracy.** The best method is in bold. Our proposed model MuRWR (marked †) shows the best performance in accuracy.

Methods	WikiVote	Slashdot	Epinions	Advogato	WN11	WN18
Random	0.497	0.500	0.493	0.340	0.090	0.078
LINE [135]	0.781	0.771	0.903	0.552	0.489	0.404
node2vec [136]	0.779	0.765	0.905	0.586	0.426	0.401
MRWR [65]	0.805	0.769	0.890	0.550	0.194	0.342
SRWR [22]	0.825	0.790	0.906	-	-	-
PRA [67]	0.813	0.804	0.913	0.683	0.580	0.556
TransE [92]	0.793	0.802	0.902	0.644	0.617	0.653
TransR [93]	0.800	0.757	0.874	0.672	0.609	0.530
MuRWR †	**0.830**	**0.820**	**0.929**	**0.727**	**0.641**	**0.689**

Table 5.4: **Performance of relation inference in terms of F1-score.** The best method is in bold. Our proposed model MuRWR (marked †) shows the best performance in F1-score.

Methods	WikiVote	Slashdot	Epinions	Advogato	WN11	WN18
Random	0.504	0.502	0.501	0.334	0.094	0.059
LINE [135]	0.524	0.592	0.706	0.479	0.204	0.202
node2vec [136]	0.519	0.583	0.673	0.532	0.256	0.206
MRWR [65]	0.703	0.661	0.809	0.698	0.149	0.131
SRWR [22]	0.742	0.730	0.822	-	-	-
PRA [67]	0.733	0.692	0.815	0.682	0.569	0.536
TransE [92]	0.725	0.687	0.821	0.648	0.571	0.642
TransR [93]	0.730	0.733	0.802	0.661	0.553	0.599
MuRWR †	**0.746**	**0.748**	**0.830**	**0.723**	**0.594**	**0.660**

macro F1-score which is a multi-class classification accuracy [137]. We repeat the above procedure 10 times, and report the average accuracy over the multiple runs for each method.

For the experiment, we select label weights w_j which provide the best inference in the validation set. To search for the label weights, we adopt the forward stepwise selection strategy [138]. Suppose we consider three weights w_1, w_2, and w_3 initialized to 1. For w_1, we fix other weights w_2 and w_3, and choose the value of w_1 which

provides the best accuracy on the validation set when varying w_1 in the range $[0, 2]$ by step size 0.2. With the selected value of w_1, we fix w_3, and repeat the above procedure for w_2. We finally perform the stepwise search for w_3 with the selected values of w_1 and w_2.

Tables 5.3 and 5.4 show the performance of MuRWR compared to other methods. Our model MuRWR is the most accurate in predicting edge labels for all the datasets in terms of accuracy. Specifically, MuRWR obtains $0.7 \sim 6.1\%$ relative improvement over the second best method. Also, the performance of MuRWR is higher than that of other methods in terms of F1-score: MuRWR outperforms the second best method by up to $0.5 \sim 5.1\%$. Note that we cannot perform relation inference using relevance scores measured by standard RWR as described in Section 5.1. These indicate that relevance scores computed by MuRWR are effective for predicting edge labels.

5.5.3 Effects of Label Weights in MuRWR

We examine the impact of label weights in MuRWR on the performance of the relation inference task. We use MuRWR-F (a version of MuRWR where all weights $\{w_j\}$ are fixed to 1) and MuRWR using the forward stepwise strategy as described above. We measure the performance of the relation inference task in terms of accuracy and F1-score. As seen in Figure 5.5, the inference performance of MuRWR is better than that of MuRWR-F in terms of accuracy and F1-score, respectively. This indicates that adjusting the label weights is helpful for the performance of MuRWR in the relation inference task.

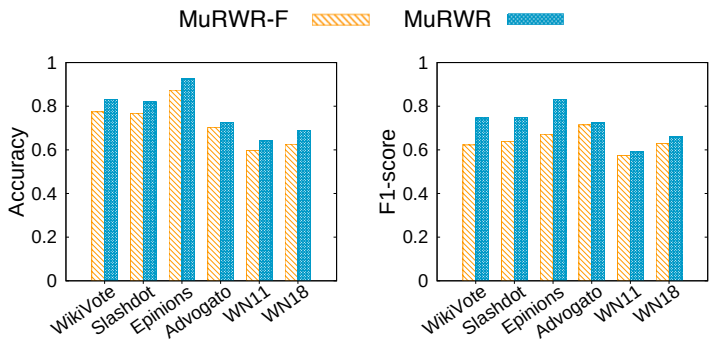
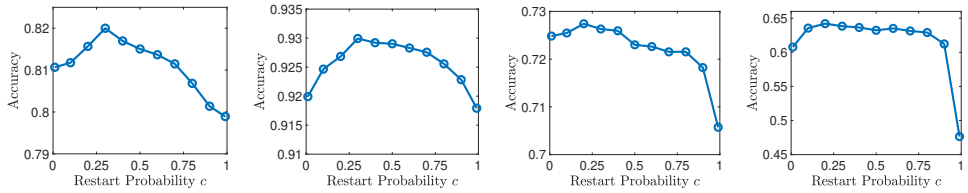


Figure 5.5: **Effect of the label weights in MuRWR.** The performance of MuRWR is better than that of MuRWR-F (i.e., MuRWR where all label weights are fixed to 1), verifying that introducing the weights improves the performance of MuRWR.

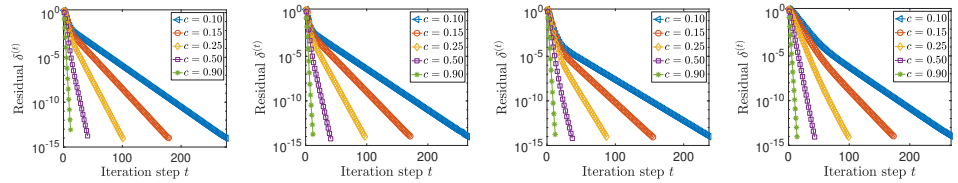
5.5.4 Effects of Restart Probability in MuRWR

We investigate the effect of restart probability c of MuRWR. We measure the inference performance of MuRWR in terms of accuracy varying c from 0.01 to 0.99. As shown in Figure 5.6, the performance of MuRWR with $c = 0.15 \sim 0.3$ is better than that of MuRWR when c is too low or high. Note that c controls how far the surfer walks from source node s. If a value of c is high, the surfer frequently jumps back to the source node; thus, the relevance score between node s and a target node t are highly affected by paths of short length from s to t, restricting the model's complexity severely. Hence, an extremely high value of c such as 0.99 has a bad influence on the performance of MuRWR. On the other hand, a too low value of c such as 0.01 also does not have a positive impact on relational reasoning since such a low value effectively ignores the source vertex. With a proper value of c between 0.15 and 0.3, MuRWR provides the best performance, and outperforms other methods as shown in Tables 5.3 and 5.4.



(a) Slashdot dataset (b) Epinions dataset (c) Advogato dataset (d) WN11 dataset

Figure 5.6: **Effect of the restart probability c in MuRWR.** The inference performance of MuRWR with $c = 0.15 \sim 0.3$ is better than that of MuRWR when c is too low or high.



(a) Slashdot dataset (b) Epinions dataset (c) Advogato dataset (d) WN11 dataset

Figure 5.7: **Convergence of MuRWR.** The residual $\delta^{(t)}$ in Algorithm 11 monotonically decreases in the datasets for $0 < c < 1$ where c is the restart probability in MuRWR.

5.5.5 Convergence of MuRWR

We explore the convergence behavior of MuRWR described in Theorem 5.1. We vary the restart probability c between 0.1 and 0.9, and measure the residual $\delta^{(t)}$ in Algorithm 11 until convergence with error tolerance $\varepsilon = 10^{-14}$. As seen in Figure 5.7, the residual $\delta^{(t)}$ monotonically decreases for $0 < c < 1$ as step t increases, and then, $\delta^{(t)}$ finally becomes less than ε. Another observation is that a high value of c accelerates the convergence rate for the residual $\delta^{(t)}$, e.g., $c = 0.9$ makes MuRWR converge faster than $c = 0.1$. The reason is that by Theorem 5.1, $\delta^{(t)} \leq 2(1-c)^t$; thus, the higher c is, the faster the residual $\delta^{(t)}$ goes toward zero. Note that an extremely high value of c such as 0.99 degrades the performance of MuRWR as shown in Figure 5.6, while a too low value of c requires many iterations to converge as shown in Figure 5.7. A value of c between 0.15 and 0.3 provides a good trade-off between inference performance and the number of iterations to converge.

5.6 Summary

We propose MuRWR (Multi-Labeled Random Walk with Restart), a novel random walk based model which accurately infers edge labels between nodes by computing relevance scores between a source node and other nodes for each edge label in edge-labeled graphs. We introduce a labeled random surfer to consider labeled edges for multi-hop relational reasoning. We provide a learning procedure based on label transitive relationships inherent in a given graph, and theoretically analyze our method including its convergence. We also show that MuRWR is a generalized version of RWR. Experimental results show MuRWR gives the highest accuracy in the relation inference task. Future works include extending the method for a graph with complex node features.

Chapter 6

Future Works

In this section, I describe future research works of this thesis. The main future research direction is to extend the approach exploiting various distinct properties inherent in real-world data to other research problems, such as pseudoinverse computation and realistic signed network generation, beyond random walk in graphs. Another promising direction is to extend the algorithms developed by this thesis to disk-based algorithms or distributed systems since the proposed methods are based on a single in-memory system. This future research will enable us to efficiently analyze much larger graphs that cannot fit in memory in a single machine.

6.1 Fast and Accurate Pseudoinverse Computation

How can we compute pseudoinverse of a sparse feature matrix efficiently and accurately for solving optimization problems? A pseudoinverse is a generalization of a matrix inverse, which has been extensively utilized as a fundamental building block for solving linear systems in machine learning. However, an approximate computation, let alone an exact computation, of pseudoinverse is very time-consuming due to its demanding time complexity, which limits it from being applied to large data.

To this end, I will develop a novel incremental singular value decomposition (SVD) based pseudoinverse method for sparse matrices. I will show that many real-world features matrices are sparse and highly skewed. Based on this observation, I will demonstrate that the non-zeros of the feature matrix can be concentrated so that

the SVD computation for pseudoinverse becomes accelerated with incremental SVD techniques [139].

6.2 Fast and Scalable Signed Network Generation

How can we efficiently generate large-scale signed networks following real-world properties? Due to its rich modeling capability of representing trust relations as positive and negative edges, signed networks have spurred much interests with various applications. Generating realistic signed networks is crucial for scalability evaluation, network simulation, and data anonymization; however, existing models for generating signed networks do not correctly reflect properties of real-world graphs.

For this problem, I have a plan to develop a fast, scalable, and parallelizable method for generating large-scale signed networks following realistic properties. I observe a self-similar balanced structure from real-world signed networks, and then simulate the self-similarity via Kronecker product to generate realistic signed networks. Also, I aim to efficiently generate very large-scale signed networks fully in parallel [140].

6.3 Disk-based Algorithms for Random Walk

Although I proposed several methods for random walk in real-world graphs, the proposed algorithms are designed for in-memory systems, i.e., an input network and intermediate data produced by each proposed method are stored only in memory. This approach has a limitation for processing very large graphs reaching tera- or peta-scale since such massive graphs cannot fit into memory. Hence, one future research direction is to extend the proposed algorithms to disk-based algorithms working on

graph databases or distributed systems.

For the purpose, I first need to carefully consider how to reduce I/O cost rather than CPU cost since the I/O cost will be the main bottleneck when it comes to designing disk-based algorithms. Therefore, one of the main technical challenges is to minimize the size of intermediate data to be stored onto disk and reduce the number of disk accesses as possible. For distributed systems, the main bottleneck could be network cost, especially when intermediate data to be transferred are very large. Thus, another challenge is how to avoid such intermediate data explosion for developing algorithms in distributed systems.

Chapter 7

Conclusion

In this dissertation, I devise fast, scalable, and exact methods, and design effective models for random walk based mining on large real-world graphs.

First, I propose BePI, a fast, memory-efficient, and scalable algorithm for random walk with restart computation on billion-scale graphs. BePI exploits hub-and-spoke structures commonly appeared in real-world graphs so that it takes the advantages of both preprocessing methods and iterative methods by incorporating an iterative method within a block elimination approach. Consequently, BePI achieves a better scalability as well as faster query time than existing methods.

Second, I design SRWR, a novel model which provides personalized rankings based on trustworthiness in signed social networks. The main idea is to introduce a signed random surfer so that it considers negative edges by changing its sign for surfing on signed networks. As a result, SRWR provides personalized trust or distrust rankings reflecting signed edges, and improves the predictive performance of various applications in signed networks. I further develop SRWR-Pre, an efficient method for computing SRWR scores in signed networks. I notice that signed networks are also sparse, and have skewed degree distribution; thus, they have the hub-and-spoke structure. SRWR-Pre exploits this structure to boost its computational performance. I demonstrate that SRWR-Pre is faster and more memory-efficient than other baseline methods.

Finally, I propose MuRWR, a novel random walk based model which accurately

infers edge labels between nodes by computing relevance scores between a source node and other nodes for each edge label in edge-labeled graphs. I introduce a labeled random surfer to consider labeled edges for multi-hop relational reasoning. I provide a learning procedure based on label transitive relationships representing logical knowledge inherent in a given graph, and theoretically analyze MuRWR w.r.t. learning and convergence. Though experiments, I show that MuRWR infers missing relations more accurately than other existing models.

As future work, I will further extend the approach exploiting distinct structures in real-world data to other research problems, such as pseudoinverse computation and synthetic graph generation, beyond random walk. Another direction is to make the proposed methods working on graphs databases (e.g., Neo4j) and distributed systems (e.g., Spark).

References

[1] L. da Fontoura Costa, O. N. O. Jr, G. Travieso, F. A. Rodrigues, P. R. V. Boas, L. Antiqueira, M. P. Viana, and L. E. C. Rocha, "Analyzing and modeling real-world phenomena with complex networks: A survey of applications," *Advances in Physics*, vol. 60, no. 3, pp. 329–412, 2011.

[2] H. Kwak, C. Lee, H. Park, and S. B. Moon, "What is twitter, a social network or a news media?," in *Proceedings of International Conference on World Wide Web (WWW 2010), Raleigh, North Carolina, USA*, pp. 591–600, 2010.

[3] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the facebook social graph," *CoRR*, vol. abs/1111.4503, 2011.

[4] H. W. Park and M. Thelwall, "Hyperlink analyses of the world wide web: A review," *Journal of Computer-Mediated Communication*, vol. 8, no. 4, p. 0, 2003.

[5] E. Garfield, "Citation indexing for studying science," *Nature*, vol. 227, no. 5259, pp. 669–671, 1970.

[6] D. Szklarczyk, A. Franceschini, S. Wyder, K. Forslund, D. Heller, J. Huerta-Cepas, M. Simonovic, A. Roth, A. Santos, and K. P. Tsafou, "String v10: Protein–protein interaction networks, integrated over the tree of life," *Nucleic Acids Research*, vol. 43, no. D1, pp. D447–D452, 2014.

[7] D. S. Bassett and E. Bullmore, "Small-world brain networks," *The Neuroscientist*, vol. 12, no. 6, pp. 512–523, 2006.

[8] J. M. Montoya and R. V. Solé, "Small world patterns in food webs," *Journal of Theoretical Biology*, vol. 214, no. 3, pp. 405–412, 2002.

[9] S. Wasserman and K. Faust, "Social network analysis in the social and behavioral sciences," *Social Network Analysis: Methods and Applications*, vol. 1994, pp. 1–27, 1994.

[10] R. D. Balicer, "Modeling infectious diseases dissemination through online role-playing games," *Epidemiology*, vol. 18, no. 2, pp. 260–261, 2007.

[11] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca, "Network analysis in the social sciences," *Science*, vol. 323, no. 5916, pp. 892–895, 2009.

[12] A.-L. Barabási, *Network Science.* Cambridge University Press, 2016.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," *Technical report, Stanford University*, 1999.

[14] L. Lovász, "Random walks on graphs: A survey," *Combinatorics*, vol. 2, no. 1, pp. 1–46, 1993.

[15] P. G. Doyle and J. L. Snell, "Random walks and electric networks," *ArXiv Preprint Math/0001057*, 2000.

[16] J. M. Kleinberg, "Hubs, authorities, and communities," *ACM Computing Surveys (CSUR)*, vol. 31, no. 4es, p. 5, 1999.

[17] G. Jeh and J. Widom, "Simrank: A measure of structural-context similarity," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2002), Edmonton, Alberta, Canada*, pp. 538–543, 2002.

[18] J. Pan, H. Yang, C. Faloutsos, and P. Duygulu, "Automatic multimedia cross-modal correlation discovery," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2004), Seattle, Washington, USA*, pp. 653–658, 2004.

[19] J. He, M. Li, H. Zhang, H. Tong, and C. Zhang, "Manifold-ranking based image retrieval," in *Proceedings of ACM International Conference on Multimedia (ICM 2004), New York, NY, USA*, pp. 9–16, 2004.

[20] H. Tong and C. Faloutsos, "Center-piece subgraphs: Problem definition and fast solutions," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2006), Philadelphia, PA, USA*, pp. 404–413, 2006.

[21] T. H. Haveliwala, "Topic-sensitive pagerank," in *Proceedings of International Conference on World Wide Web (WWW 2002), Honolulu, Hawaii, USA*, pp. 517–526, 2002.

[22] J. Jung, W. Jin, L. Sael, and U. Kang, "Personalized ranking in signed networks using signed random walk with restart," in *Proceedings of IEEE International Conference on Data Mining (ICDM 2016), Barcelona, Spain*, pp. 973–978, 2016.

[23] S. Lee, S. Park, M. Kahng, and S. Lee, "Pathrank: A novel node ranking measure on a heterogeneous graph for recommender systems," in *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM 2012), Maui, HI, USA*, pp. 1637–1641, 2012.

[24] M. Nykl, M. Campr, and K. Jezek, "Author ranking based on personalized pagerank," *Journal of Informetrics*, vol. 9, no. 4, pp. 777–799, 2015.

[25] H. Sayyadi and L. Getoor, "Futurerank: Ranking scientific articles by predicting their future pagerank," in *Proceedings of the SIAM International Conference on Data Mining (SDM 2009), Sparks, Nevada, USA*, pp. 533–544, 2009.

[26] W. Jin, J. Jung, and U. Kang, "Supervised and extended restart in random walks for ranking and link prediction in networks," *PloS one*, vol. 14, no. 3, p. e0213857, 2019.

[27] L. Backstrom and J. Leskovec, "Supervised random walks: Predicting and recommending links in social networks," in *Proceedings of ACM International Conference on Web Search and Web Data Mining (WSDM 2011), Hong Kong, China*, pp. 635–644, 2011.

[28] W. Liu and L. Lü, "Link prediction based on local random walk," *EPL (Europhysics Letters)*, vol. 89, no. 5, p. 58007, 2010.

[29] D. Liben-Nowell and J. M. Kleinberg, "The link prediction problem for social networks," in *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM 2003), New Orleans, Louisiana, USA*, pp. 556–559, 2003.

[30] L. Li, Y. Yao, J. Tang, W. Fan, and H. Tong, "QUINT: on query-specific optimal networks," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2016), San Francisco, CA, USA*, pp. 985–994, 2016.

[31] X. Yu, Q. Gu, M. Zhou, and J. Han, "Citation prediction in heterogeneous bibliographic networks," in *Proceedings of SIAM International Conference on Data Mining (SDM 2012), Anaheim, California, USA, April 26-28, 2012*, pp. 1119–1130, 2012.

[32] H. Park, J. Jung, and U. Kang, "A comparative study of matrix factorization and random walk with restart in recommender systems," in *Proceedings of IEEE International Conference on Big Data (BigData 2017), Boston, MA, USA*, pp. 756–765, 2017.

[33] Z. Jiang, H. Liu, B. Fu, Z. Wu, and T. Zhang, "Recommendation in heterogeneous information networks based on generalized random walk model and bayesian personalized ranking," in *Proceedings of ACM International Conference on Web Search and Data Mining (WSDM 2018), Marina Del Rey, CA, USA*, pp. 288–296, 2018.

[34] P. Gupta, A. Goel, J. J. Lin, A. Sharma, D. Wang, and R. Zadeh, "WTF: the who to follow service at twitter," in *Proceedings of International World Wide Web Conference (WWW 2013), Rio de Janeiro, Brazil*, pp. 505–514, 2013.

[35] R. Baral and T. Li, "MAPS: A multi aspect personalized POI recommender system," in *Proceedings of ACM International Conference on Recommender Systems (RecSys 2016), Boston, MA, USA*, pp. 281–284, 2016.

[36] C. Faloutsos, K. S. McCurley, and A. Tomkins, "Fast discovery of connection subgraphs," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2004), Seattle, Washington, USA*, pp. 118–127, 2004.

[37] H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad, "Fast best-effort pattern matching in large attributed graphs," in *Proceedings of ACM International*

Conference on Knowledge Discovery and Data Mining (KDD 2007), San Jose, California, USA, pp. 737–746, 2007.

[38] D. Shahaf and C. Guestrin, "Connecting the dots between news articles," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2010), Washington, DC, USA*, pp. 623–632, 2010.

[39] G. Kasneci, S. Elbassuoni, and G. Weikum, "MING: mining informative entity relationship subgraphs," in *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM 2009), Hong Kong, China*, pp. 1653–1656, 2009.

[40] R. Pienta, A. Tamersoy, H. Tong, and D. H. Chau, "MAGE: matching approximate patterns in richly-attributed graphs," in *Proceedings of IEEE International Conference on Big Data (BigData 2014), Washington, DC, USA*, pp. 585–590, 2014.

[41] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos, "Neighborhood formation and anomaly detection in bipartite graphs," in *Proceedings of IEEE International Conference on Data Mining (ICDM 2005), Houston, Texas, USA*, pp. 418–425, 2005.

[42] P. Chirita, J. Diederich, and W. Nejdl, "Mailrank: using ranking for spam detection," in *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM 2005), Bremen, Germany*, pp. 373–380, 2005.

[43] Z. Gyöngyi, H. Garcia-Molina, and J. O. Pedersen, "Combating web spam with trustrank," in *Proceedings of International Conference on Very Large Data Bases (VLDB 2004), Toronto, Canada*, pp. 576–587, 2004.

[44] N. Chiluka, N. Andrade, D. Gkorou, and J. A. Pouwelse, "Personalizing eigentrust in the face of communities and centrality attack," in *Proceedings of IEEE International Conference on Advanced Information Networking and Applications (AINA 2012), Fukuoka, Japan*, pp. 503–510, 2012.

[45] D. Eswaran and C. Faloutsos, "Sedanspot: Detecting anomalies in edge streams," in *Proceedings of IEEE International Conference on Data Mining (ICDM 2018), Singapore*, pp. 953–958, 2018.

[46] Z. Yao, P. Mark, and M. Rabbat, "Anomaly detection using proximity graph and pagerank algorithm," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 7, no. 4, pp. 1288–1300, 2012.

[47] H. Avron and L. Horesh, "Community detection using time-dependent personalized pagerank," in *Proceedings of International Conference on Machine Learning (ICML 2015), Lille, France*, pp. 1795–1803, 2015.

[48] R. Andersen, F. R. K. Chung, and K. J. Lang, "Local graph partitioning using pagerank vectors," in *Proceedings of IEEE Symposium on Foundations of Computer Science (FOCS 2006), Berkeley, California*, pp. 475–486, 2006.

[49] J. J. Whang, D. F. Gleich, and I. S. Dhillon, "Overlapping community detection using seed set expansion," in *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM 2013), San Francisco, CA, USA*, pp. 2099–2108, 2013.

[50] D. F. Gleich and C. Seshadhri, "Vertex neighborhoods, low conductance cuts, and good seeds for local community methods," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2012), Beijing, China*, pp. 597–605, 2012.

[51] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," *PVLDB*, vol. 2, no. 1, pp. 718–729, 2009.

[52] K. Macropol, T. Can, and A. K. Singh, "RRW: repeated random walks on genome-scale protein networks for local cluster discovery," *BMC Bioinformatics*, vol. 10, p. 283, 2009.

[53] Y. Saad and M. H. Schultz, "Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *Journal on Scientific and Statistical Computing*, vol. 7, no. 3, pp. 856–869, 1986.

[54] K. Shin, J. Jung, L. Sael, and U. Kang, "BEAR: block elimination approach for random walk with restart on large graphs," in *Proceedings of ACM International Conference on Management of Data (SIGMOD 2015), Melbourne, Victoria*, pp. 1571–1585, 2015.

[55] D. F. Gleich and M. Polito, "Approximating personalized pagerank with minimal use of web graph data," *Internet Mathematics*, vol. 3, no. 3, pp. 257–294, 2007.

[56] H. Tong, C. Faloutsos, and J. Pan, "Fast random walk with restart and its applications," in *Proceedings of IEEE International Conference on Data Mining (ICDM 2006), Hong Kong, China*, pp. 613–622, 2006.

[57] P. Lofgren, S. Banerjee, A. Goel, and S. Comandur, "FAST-PPR: scaling personalized pagerank estimation for large graphs," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2014), New York, NY, USA*, pp. 1436–1445, 2014.

[58] W. Xie, D. Bindel, A. J. Demers, and J. Gehrke, "Edge-weighted personalized pagerank: Breaking A decade-old performance barrier," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2015), Sydney, NSW, Australia*, pp. 1325–1334, 2015.

[59] A. W. Yu, N. Mamoulis, and H. Su, "Reverse top-k search using random walk with restart," *PVLDB*, vol. 7, no. 5, pp. 401–412, 2014.

[60] Y. Fujiwara, M. Nakatsuji, M. Onizuka, and M. Kitsuregawa, "Fast and exact top-k search for random walk with restart," *PVLDB*, vol. 5, no. 5, pp. 442–453, 2012.

[61] Y. Wu, R. Jin, and X. Zhang, "Fast and unified local search for random walk based k-nearest-neighbor query in large graphs," in *Proceedings of ACM International Conference on Management of Data (SIGMOD 2014), Snowbird, UT, USA*, pp. 1139–1150, 2014.

[62] M. S. Gupta, A. Pathak, and S. Chakrabarti, "Fast algorithms for topk personalized pagerank queries," in *Proceedings of International Conference on World Wide Web (WWW 2008), Beijing, China*, pp. 1225–1226, 2008.

[63] J. Kunegis, A. Lommatzsch, and C. Bauckhage, "The slashdot zoo: Mining a social network with negative edges," in *Proceedings of International Conference on World Wide Web (WWW 2009), Madrid, Spain*, pp. 741–750, 2009.

[64] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proceedings of the ACM International Conference on Management of Data (SIGMOD 2008), Vancouver, BC, Canada*, pp. 1247–1250, 2008.

[65] M. Shahriari and M. Jalili, "Ranking nodes in signed social networks," *Social Network Analysis and Mining*, vol. 4, no. 1, p. 172, 2014.

[66] Z. Wu, C. C. Aggarwal, and J. Sun, "The troll-trust model for ranking in signed networks," in *Proceedings of ACM International Conference on Web Search and Data Mining (WSDM 2016), San Francisco, CA, USA*, pp. 447–456, 2016.

[67] N. Lao, T. M. Mitchell, and W. W. Cohen, "Random walk inference and learning in A large scale knowledge base," in *Proceedings of International Conference on Empirical Methods in Natural Language Processing (EMNLP 2011), Edinburgh, UK*, pp. 529–539, 2011.

[68] S. Wang, R. Yang, X. Xiao, Z. Wei, and Y. Yang, "FORA: simple and effective approximate single-source personalized pagerank," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2017), Halifax, NS, Canada*, pp. 505–514, 2017.

[69] U. Kang, C. E. Tsourakakis, and C. Faloutsos, "PEGASUS: A peta-scale graph mining system," in *Proceedings of IEEE International Conference on Data Mining (ICDM 2009), Miami, Florida, USA*, pp. 229–238, 2009.

[70] H. Park, C. Park, and U. Kang, "Pegasusn: A scalable and versatile graph mining system," in *Proceedings of AAAI Conference on Artificial Intelligence (AAAI 2018), New Orleans, Louisiana, USA*, pp. 8214–8215, 2018.

[71] J. E. Gonzalez, R. S. Xin, A. Dave, D. Crankshaw, M. J. Franklin, and I. Stoica, "Graphx: Graph processing in a distributed dataflow framework," in *Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI 2014), Broomfield, CO, USA*, pp. 599–613, 2014.

[72] Y. Bu, V. R. Borkar, J. Jia, M. J. Carey, and T. Condie, "Pregelix: Big(ger) graph analytics on a dataflow engine," *PVLDB*, vol. 8, no. 2, pp. 161–172, 2014.

[73] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *Proceedings of ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM 1999), Cambridge, Massachusetts, USA*, pp. 251–262, 1999.

[74] D. Cartwright and F. Harary, "Structural balance: a generalization of heider's theory.," *Psychological Review*, vol. 63, no. 5, p. 277, 1956.

[75] F. Heider, "Attitudes and cognitive organization," *The Journal of Psychology*, vol. 21, no. 1, pp. 107–112, 1946.

[76] J. Jung, K. Shin, L. Sael, and U. Kang, "Random walk with restart on large graphs using block elimination," *ACM Transactions on Database Systems (TODS)*, vol. 41, no. 2, pp. 12:1–12:43, 2016.

[77] Y. Saad, "A flexible inner-outer preconditioned GMRES algorithm," *SIAM Journal of Scientific Computing*, vol. 14, no. 2, pp. 461–469, 1993.

[78] L. N. Trefethen and D. Bau, *Numerical Linear Algebra.* SIAM, 1997.

[79] Y. Saad, *Iterative Methods for Sparse Linear Systems.* SIAM, 2003.

[80] J. Jung, N. Park, L. Sael, and U. Kang, "Bepi: Fast and memory-efficient method for billion-scale random walk with restart," in *Proceedings of ACM International Conference on Management of Data (SIGMOD 2017), Chicago, Illinois, USA*, pp. 789–804, 2017.

[81] J. Jung, W. Jin, and U. Kang, "Random walk-based ranking in signed social networks: Model and algorithms," *Knowledge and Information Systems (KAIS)*, pp. 1–40, 2019.

[82] H. Tong, C. Faloutsos, and J. Pan, "Random walk with restart: Fast solutions and applications," *Knowledge and Information Systems (KAIS)*, vol. 14, no. 3, pp. 327–346, 2008.

[83] A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond: the Science of Search Engine Rankings.* Princeton University Press, 2011.

[84] Y. Fujiwara, M. Nakatsuji, T. Yamamuro, H. Shiokawa, and M. Onizuka, "Efficient personalized pagerank with accuracy assurance," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2012), Beijing, China*, pp. 15–23, 2012.

[85] B. Bahmani, K. Chakrabarti, and D. Xin, "Fast personalized pagerank on mapreduce," in *Proceedings of ACM International Conference on Management of Data (SIGMOD 2011), Athens, Greece*, pp. 973–984, 2011.

[86] T. H. Kim, K. M. Lee, and S. U. Lee, "Generative image segmentation using random walks with restart," in *Proceedings of European Conference on Computer Vision (ECCV 2008), Marseille, France*, pp. 264–275, 2008.

[87] S. Zhu, L. Zou, and B. Fang, "Content based image retrieval via a transductive model," *J. Intell. Inf. Syst.*, vol. 42, no. 1, pp. 95–109, 2014.

[88] Z. Yin, M. Gupta, T. Weninger, and J. Han, "A unified framework for link recommendation using random walks," in *Proceedings of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2010), Odense, Denmark*, pp. 152–159, 2010.

[89] D. Song and D. A. Meyer, "Recommending positive links in signed social networks by optimizing a generalized AUC," in *Proceedings of AAAI International Conference on Artificial Intelligence, Austin, Texas, USA*, pp. 290–296, 2015.

[90] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg, "Predicting positive and negative links in online social networks," in *Proceedings of International Conference on World Wide Web (WWW 2010), Raleigh, North Carolina, USA*, pp. 641–650, 2010.

[91] R. V. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *Proceedings of International Conference on World Wide Web (WWW 2004), New York, NY, USA*, pp. 403–412, 2004.

[92] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proceedings of Advances in Neural Information Processing Systems (NIPS 2013), Lake Tahoe, Nevada, USA*, pp. 2787–2795, 2013.

[93] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proceedings of AAAI International Conference on Artificial Intelligence (AAAI 2015), Austin, Texas, USA*, pp. 2181–2187, 2015.

[94] I. Antonellis, H. Garcia-Molina, and C. Chang, "Simrank++: Query rewriting through link analysis of the click graph," *PVLDB*, vol. 1, no. 1, pp. 408–421, 2008.

[95] S. Chakrabarti, A. Pathak, and M. Gupta, "Index design and query processing for graph conductance search," *VLDB Journal*, vol. 20, no. 3, pp. 445–470, 2011.

[96] S. Lee, S. Song, M. Kahng, D. Lee, and S. Lee, "Random walk based entity ranking on graph for multidimensional recommendation," in *Proceedings of ACM International Conference on Recommender Systems (RecSys 2011), Chicago, IL, USA, October 23-27, 2011*, pp. 93–100, 2011.

[97] K. Kim, J. Jung, J. Ryu, H. Park, J. P. Joohee, S. Jeong, U. Kang, and S. Myaeng, "A new question answering approach with conceptual graphs," in *COnférence en Recherche d'Informations et Applications - French Information Retrieval Conference (CORIA 2017), Marseille, France*, pp. 218–234, 2017.

[98] U. Kang and C. Faloutsos, "Beyond 'caveman communities': Hubs and spokes for graph compression and mining," in *Proceedings of IEEE International Conference on Data Mining (ICDM 2011), Vancouver, BC, Canada*, pp. 300–309, 2011.

[99] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2009.

[100] I. S. Duff, R. G. Grimes, and J. G. Lewis, "Sparse matrix test problems," *ACM Transactions on Mathematical Software (TOMS)*, vol. 15, no. 1, pp. 1–14, 1989.

[101] A. N. Langville and C. D. Meyer, "A reordering for the pagerank problem," *SIAM Journal of Scientific Computing*, vol. 27, no. 6, pp. 2112–2120, 2006.

[102] F. Zhang, *The Schur Complement and Its Applications*, vol. 4. Springer Science & Business Media, 2006.

[103] M. Benzi, "Preconditioning techniques for large linear systems: A survey," *Journal of Computational Physics*, vol. 182, no. 2, pp. 418–477, 2002.

[104] M. Benzi, C. D. Meyer, and M. Tuma, "A sparse approximate inverse preconditioner for the conjugate gradient method," *SIAM Journal of Scientific Computing*, vol. 17, no. 5, pp. 1135–1149, 1996.

[105] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra.* SIAM, 2000.

[106] M. Szell, R. Lambiotte, and S. Thurner, "Multirelational organization of large-scale social networks in an online world," *Proceedings of the National Academy of Sciences*, vol. 107, no. 31, pp. 13636–13641, 2010.

[107] B. Yang, W. K. Cheung, and J. Liu, "Community mining from signed social networks," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 19, no. 10, pp. 1333–1348, 2007.

[108] M. Yoon, W. Jin, and U. Kang, "Fast and accurate random walk with restart on dynamic graphs with guarantees," in *Proceedings of International Conference on World Wide Web (WWW 2018), Lyon, France*, pp. 409–418, 2018.

[109] M. Yoon, J. Jung, and U. Kang, "TPA: fast, scalable, and accurate method for approximate random walk with restart on billion scale graphs," in *Proceedings of IEEE International Conference on Data Engineering (ICDE 2018), Paris, France*, pp. 1132–1143, 2018.

[110] A. Mishra and A. Bhattacharya, "Finding the bias and prestige of nodes in networks based on trust scores," in *Proceedings of International Conference on World Wide Web (WWW 2011), Hyderabad, India*, pp. 567–576, 2011.

[111] M. E. Taylor, *Measure Theory and Integration.* American Mathematical Society, 2006.

[112] D. A. Easley and J. M. Kleinberg, *Networks, Crowds, and Markets - Reasoning About a Highly Connected World.* Cambridge University Press, 2010.

[113] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg, "Signed networks in social media," in *Proceedings of International Conference on Human Factors in Computing Systems (CHI 2010), Atlanta, Georgia, USA*, pp. 1361–1370, 2010.

[114] J. A. Davis, "Clustering and structural balance in graphs," *Human Relations*, vol. 20, no. 2, pp. 181–187, 1967.

[115] G. Strang, *Linear Algebra and Its Applications*. Thomson, Brooks/Cole, 2006.

[116] C. F. V. Loan, "Matrix computations (johns hopkins studies in mathematical sciences)," 1996.

[117] Y. Lim, U. Kang, and C. Faloutsos, "Slashburn: Graph compression and mining beyond caveman communities," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 26, no. 12, pp. 3077–3089, 2014.

[118] A. X. Zheng, A. Y. Ng, and M. I. Jordan, "Stable algorithms for link analysis," in *Proceedings of ACM International Conference on Research and Development in Information Retrieval (SIGIR 2001), New Orleans, Louisiana, USA*, pp. 258–266, 2001.

[119] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[120] B. Perozzi, M. Schueppert, J. Saalweachter, and M. Thakur, "When recommendation goes wrong: Anomalous link discovery in recommendation networks," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2016), San Francisco, CA, USA*, pp. 569–578, 2016.

[121] F. Bonchi, A. Gionis, F. Gullo, and A. Ukkonen, "Distance oracles in edge-labeled graphs," in *Proceedings of International Conference on Extending Database Technology (EDBT 2014), Athens, Greece*, pp. 547–558, 2014.

[122] G. A. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[123] J. J. McAuley and J. Leskovec, "Learning to discover social circles in ego networks," in *Proceedings of Advances in Neural Information Processing Systems (NIPS 2012), Lake Tahoe, Nevada, USA*, pp. 548–556, 2012.

[124] D. Eswaran, S. Günnemann, C. Faloutsos, D. Makhija, and M. Kumar, "Zoobp: Belief propagation for heterogeneous networks," *PVLDB*, vol. 10, no. 5, pp. 625–636, 2017.

[125] D. Koutra, T. Ke, U. Kang, D. H. Chau, H. K. Pao, and C. Faloutsos, "Unifying guilt-by-association approaches: Theorems and fast algorithms," in *Proceedings of Machine Learning and Knowledge Discovery in Databases - European Conference (ECML–PKDD 2011), Athens, Greece*, pp. 245–260, 2011.

[126] M. Latapy, "Main-memory triangle computations for very large (sparse (power-law)) graphs," *Theoretical Computer Science*, vol. 407, no. 1-3, pp. 458–473, 2008.

[127] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2014), New York, NY, USA*, pp. 601–610, 2014.

[128] X. Wang, C. V. Eeden, and J. V. Zidek, "Asymptotic properties of maximum weighted likelihood estimators," *Journal of Statistical Planning and Inference*, vol. 119, no. 1, pp. 37–54, 2004.

[129] A. J. Laub, *Matrix Analysis - for Scientists and Engineers*. SIAM, 2005.

[130] P. Massa and P. Avesani, "Controversial users demand local trust metrics: An experimental study on epinions.com community," in *Proceedings of AAAI International Conference on Artificial Intelligence (AAAI 2005), Pittsburgh, Pennsylvania, USA*, pp. 121–126, 2005.

[131] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Proceedings of Advances in Neural Information Processing Systems (NIPS 2013), Lake Tahoe, Nevada, USA*, pp. 926–934, 2013.

[132] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data - application to word-sense disambiguation," *Machine Learning*, vol. 94, no. 2, pp. 233–259, 2014.

[133] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg, "Governance in social media: A case study of the wikipedia promotion process," in *Proceedings of International Conference on Weblogs and Social Media (ICWSM 2010), Washington, DC, USA*, 2010.

[134] P. Massa, M. Salvetti, and D. Tomasoni, "Bowling alone and trust decline in social network sites," in *Proceedings of IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2009), Chengdu, China*, pp. 658–663, 2009.

[135] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: large-scale information network embedding," in *Proceedings of International Conference on World Wide Web (WWW 2015), Florence, Italy*, pp. 1067–1077, 2015.

[136] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (KDD 2016), San Francisco, CA, USA*, pp. 855–864, 2016.

[137] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.

[138] K. Murphy, "Machine learning: A probabilistic approach," *Massachusetts Institute of Technology*, pp. 1–21, 2012.

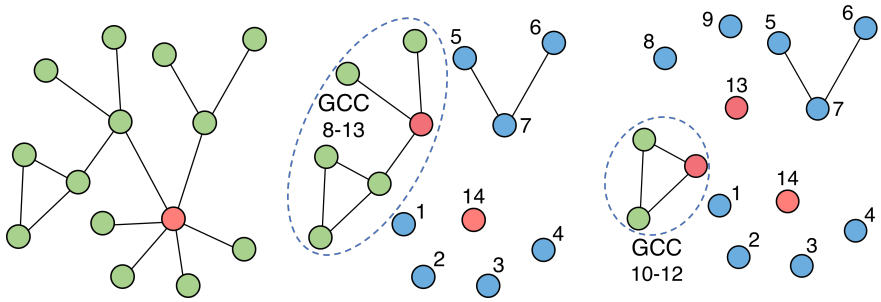
[139] J. Jang, D. Choi, J. Jung, and U. Kang, "Zoom-svd: Fast and memory efficient method for extracting key patterns in an arbitrary time range," in *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM 2018), Torino, Italy*, pp. 1083–1092, 2018.

[140] J. Jung, H.-M. Park, and U. Kang, "Balansing: Fast and scalable generation of realistic signed networks," in *Proceedings of International Conference on Extending Database Technology (EDBT 2020), Copenhagen, Denmark*, 2020.

[141] J. Demmel, *Applied Numerical Linear Algebra.* SIAM, 1997.

Appendix

A.1 Hub-and-Spoke Reordering Method



(a) Step 1 (b) Step 2 (c) Step 3

Figure A.1: **Node reordering based on hub-and-spoke method** when $\lceil kn \rceil = 1$ where $\lceil kn \rceil$ indicates the number of selected hubs at each step, and k is the hub selection ratio ($0 < k < 1$). Red nodes are hubs; blue nodes are spokes that belong to the disconnected components; green colored are nodes that belong to the giant connected component. At Step 1 in (a), the method disconnects a hub node, and assigns node ids as shown in (b). The hub node gets the highest id (14), the spoke nodes get the lowest ids ($1 \sim 7$), and the GCC gets the middle ids ($8 \sim 13$). The next iteration starts on the GCC in (b), and the node ids are assigned as in (c)

SlashBurn [98, 117] is a node reordering algorithm which concentrates non-zero entries of the adjacency matrix of a given graph based on the hub-and-spoke structure. Let n be the number of nodes in a graph, and k be the hub selection ratio whose range is between 0 and 1 where $\lceil kn \rceil$ indicates the number of nodes selected by SlashBurn as hubs. For each iteration, SlashBurn disconnects $\lceil kn \rceil$ high degree nodes, called *hub nodes*, from the graph; then the graph is split into the giant connected component (GCC) and the disconnected components. The nodes in the disconnected components are called *spokes*, and each disconnected component forms a block in the matrix as in Figures 3.2(c) and 4.6. Then, SlashBurn reorders nodes such that the hub nodes get the highest ids, the spokes get the lowest ids, and the nodes in the GCC get

the ids in the middle. SlashBurn repeats this procedure on the GCC recursively until the size of GCC becomes smaller than $\lceil kn \rceil$. After SlashBurn is done, the reordered adjacency matrix contains a large and sparse block diagonal matrix in the upper left area, as shown in Figures 3.2(c) and 4.6. Figure A.1 depicts the procedure of SlashBurn when $\lceil kn \rceil = 1$.

A.2 Time Complexity of Sparse Matrix Multiplication

Lemma A.1 (Sparse Matrix Multiplication [79])**.** *Suppose that* $\mathbf{A}$ *and* $\mathbf{B}$ *are* $p \times q$ *and* $q \times r$ *sparse matrices, respectively, and* $\mathbf{A}$ *has* $\mathrm{nnz}(\mathbf{A})$ *non-zeros. Calculating* $\mathbf{C} = \mathbf{AB}$ *using sparse matrix multiplication requires* $O(\mathrm{nnz}(\mathbf{A})r)$.

A.3 Details of Preconditioned GMRES

Preconditioned GMRES [77, 78] computes the solution $\mathbf{r}_2$ of the preconditioned linear system in Equation (3.9), $\mathbf{M}^{-1}\mathbf{S}\mathbf{r}_2 = \mathbf{M}^{-1}\tilde{\mathbf{q}}_2$, where $\mathbf{S}$ is incomplete LU decomposed into $\tilde{\mathbf{L}}_2$ and $\tilde{\mathbf{U}}_2$, and $\mathbf{M}^{-1} = \tilde{\mathbf{U}}_2^{-1}\tilde{\mathbf{L}}_2^{-1}$ is a preconditioner. Algorithm 12 describes the procedure of preconditioned GMRES. It iteratively finds $\mathbf{r}_2^{(i)}$ which minimizes the residual $\|\mathbf{M}^{-1}(\mathbf{S}\mathbf{r}_2^{(i)} - \tilde{\mathbf{q}}_2)\|_2$ such that $\mathbf{r}_2^{(i)}$ is in the i-th order preconditioned Krylov subspace $\mathcal{K}_i$ represented as follows:

$$\mathcal{K}_i = \{\mathbf{M}^{-1}\tilde{\mathbf{q}}_2, (\mathbf{M}^{-1}\mathbf{S})\mathbf{M}^{-1}\tilde{\mathbf{q}}_2, ..., (\mathbf{M}^{-1}\mathbf{S})^{i-1}\mathbf{M}^{-1}\tilde{\mathbf{q}}_2\}.$$

Since the vectors consisting of the Krylov subspace $\mathcal{K}_i$ are almost linearly dependent, directly finding the solution using the vectors as basis could be unstable.

Algorithm 12: Preconditioned GMRES [77, 78]

Input: matrix: $\mathbf{S}$, vector: $\tilde{\mathbf{q}}_2$, preconditioner: $\tilde{\mathbf{L}}_2$ and $\tilde{\mathbf{U}}_2$, error tolerance: ε

Output: solution: $\mathbf{r}_2$ of the preconditioned system in Equation (3.9)

1: $\mathbf{t} = \tilde{\mathbf{U}}_2^{-1}(\tilde{\mathbf{L}}_2^{-1}\tilde{\mathbf{q}}_2) \Leftrightarrow \tilde{\mathbf{U}}_2 \backslash_B (\tilde{\mathbf{L}}_2 \backslash_F \tilde{\mathbf{q}}_2)$
2: $\mathbf{p}_1 = \mathbf{t}/\|\mathbf{t}\|_2$
3: **for** i=1:n_2 **do**
   find an orthonormal vector $\mathbf{p}_{i+1}$ *against* $\{\mathbf{p}_1, ..., \mathbf{p}_i\}$ *in* $\mathcal{K}_{i+1}$ *using the following iteration (lines 4∼10), called Arnoldi Iteration [78]*
4: $\mathbf{v} = \tilde{\mathbf{U}}_2^{-1}(\tilde{\mathbf{L}}_2^{-1}(\mathbf{S}\mathbf{p}_i)) \Leftrightarrow \tilde{\mathbf{U}}_2 \backslash_B (\tilde{\mathbf{L}}_2 \backslash_F (\mathbf{S}\mathbf{p}_i))$
5: **for** j=1:i **do**
6: $h_{j,i} = \mathbf{p}_j^T \mathbf{v}$
7: $\mathbf{v} = \mathbf{v} - h_{j,i}\mathbf{p}_j$
8: **end for**
9: $h_{i+1,i} = \|\mathbf{v}\|_2$
10: $\mathbf{p}_{i+1} = \mathbf{v}/h_{i+1,i}$
11: solve $\mathbf{y}^* \leftarrow \text{argmin}_{\mathbf{y}} \|\mathbf{H}_i\mathbf{y} - \|\mathbf{t}\|_2\mathbf{e}_1\|_2$ using a linear least square method (e.g., QR decomposition)
12: $\mathbf{r}_2^{(i)} = \mathbf{P}_i\mathbf{y}^*$
13: **if** $\|\mathbf{H}_i\mathbf{y}^* - \|\mathbf{t}\|_2\mathbf{e}_1\|_2 < \varepsilon$ **then**
14: $\mathbf{r}_2 \leftarrow \mathbf{r}_2^{(i)}$; break
15: **end if**
16: **end for**
17: **return** $\mathbf{r}_2$

Instead, preconditioned GMRES finds $\mathbf{r}_2^{(i)}$ in a subspace generated by the orthonormal vectors $\{\mathbf{p}_1, ..., \mathbf{p}_i\}$ in $\mathcal{K}_i$, which are iteratively computed by Arnoldi Iteration (lines 4∼10). The solution $\mathbf{r}_2^{(i)}$ is $\mathbf{P}_i\mathbf{y}$ where $\mathbf{P}_i = \left[\mathbf{p}_1, ..., \mathbf{p}_i\right]$ is an orthonormal matrix and $\mathbf{y} \in \mathbb{R}^n$. Then, the partial similarity transformation of $\mathbf{M}^{-1}\mathbf{S}$ by Arnoldi Iteration at the i-th iteration is represented as follows:

$$(\mathbf{M}^{-1}\mathbf{S})\mathbf{P}_i = \mathbf{P}_{i+1}\mathbf{H}_i$$

where $\mathbf{H}_i$ is a Hessenberg matrix in $\mathbb{R}^{(i+1)\times i}$ and $h_{j,k}$ is the (j,k)-th entry of $\mathbf{H}_i$. Since the columns of $\mathbf{P}_i$ are orthonormal (i.e., $\mathbf{P^T}_i\mathbf{P}_i = \mathbf{I}$) and $\mathbf{r}_2^{(i)} = \mathbf{P}_i\mathbf{y}$, the original residual

is modified as follows:

$$\|\mathbf{M}^{-1}(\mathbf{S}\mathbf{r}_2^{(i)} - \tilde{\mathbf{q}}_2)\|_2 \Rightarrow \|\mathbf{P}_{i+1}\mathbf{H}_i\mathbf{y} - \mathbf{M}^{-1}\tilde{\mathbf{q}}_2\|_2$$
$$\Rightarrow \|\mathbf{H}_i\mathbf{y} - \mathbf{P}_{i+1}^T\mathbf{M}^{-1}\tilde{\mathbf{q}}_2\|_2 \Rightarrow \|\mathbf{H}_i\mathbf{y} - \|\mathbf{M}^{-1}\tilde{\mathbf{q}}_2\|_2\mathbf{e}_1\|_2$$

Note that we exploit $\mathbf{p}_1 = (\mathbf{M}^{-1}\tilde{\mathbf{q}}_2)/\|\mathbf{M}^{-1}\tilde{\mathbf{q}}_2\|_2$ (lines 1∼2) for the last transformation. Consequently, preconditioned GMRES finds $\mathbf{y}^*$ which minimizes the modified residual $\|\mathbf{H}_i\mathbf{y} - \|\mathbf{M}^{-1}\tilde{\mathbf{q}}_2\|_2\mathbf{e}_1\|_2$ using a linear least square method such as QR decomposition (line 11); and then, it computes the solution $\mathbf{r}_2^{(i)}$ based on $\mathbf{y}^*$ (line 12). Preconditioned GMRES repeats the procedure for finding $\mathbf{r}_2^{(i)}$ until the residual is less than ε (line 13).

Note that we do not need to obtain $\mathbf{M}^{-1} = \tilde{\mathbf{U}}_2^{-1}\tilde{\mathbf{L}}_2^{-1}$ if $\mathbf{M}$ consists of triangular matrices. For example, when we should perform an operation in the form of $\mathbf{z} = \tilde{\mathbf{U}}_2^{-1}(\tilde{\mathbf{L}}_2^{-1}\mathbf{w})$ (lines 1 and 4), forward and backward substitutions efficiently compute $\mathbf{z}$ without matrix inversion [141], i.e., $\mathbf{z} = \tilde{\mathbf{U}}_2\backslash_B(\tilde{\mathbf{L}}_2\backslash_F\mathbf{w})$ where $\backslash_F$ and $\backslash_B$ are defined as follows:

- Forward substitution $\backslash_F$: $\mathbf{x} = \mathbf{L}^{-1}\mathbf{b} \Leftrightarrow \mathbf{x} = \mathbf{L}\backslash_F\mathbf{b}$ where $\mathbf{L}$ is a lower triangular matrix.
- Backward substitution $\backslash_B$: $\mathbf{x} = \mathbf{U}^{-1}\mathbf{b} \Leftrightarrow \mathbf{x} = \mathbf{U}\backslash_B\mathbf{b}$ where $\mathbf{U}$ is an upper triangular matrix.

The time complexity of the substitution algorithms is the same as that of matrix-vector multiplication [141]. Hence, preconditioned GMRES efficiently finds $\mathbf{r}_2$ of the preconditioned system without inverting $\tilde{\mathbf{L}}_2$ and $\tilde{\mathbf{U}}_2$.

A.4 Detailed Description of Evaluation Metrics

We describe the details of metrics used in the link prediction and the troll identification tasks. The metrics for the sign prediction task is described in Section 4.4.5.

A.4.1 Link Prediction

- GAUC (Generalized AUC): Song et al. [89] proposed GAUC which measures the quality of link prediction in signed networks. An ideal personalized ranking w.r.t. a seed node s needs to rank nodes with positive links to s at the top, those with negative links at the bottom, and other unknown status nodes in the middle of the ranking. For a seed node s, suppose that $\mathbf{P}_s$ is the set of positive nodes potentially connected by s, $\mathbf{N}_s$ is that of negative nodes, and $\mathbf{O}_s$ is that of the other nodes. Then, GAUC of the personalized ranking w.r.t. s is defined as follows:

$$\begin{aligned}\mathrm{GAUC}_s = &\frac{\eta}{|\mathbf{P}_s|(|\mathbf{O}_s|+|\mathbf{N}_s|)}\left(\sum_{p\in\mathbf{P}_s}\sum_{i\in\mathbf{O}_s\cup\mathbf{N}_s}\mathbb{I}(\mathbf{r}_p>\mathbf{r}_i)\right)\\ &+\frac{1-\eta}{|\mathbf{N}_s|(|\mathbf{O}_s|+|\mathbf{P}_s|)}\left(\sum_{i\in\mathbf{O}_s\cup\mathbf{P}_s}\sum_{n\in\mathbf{N}_s}\mathbb{I}(\mathbf{r}_i<\mathbf{r}_n)\right)\end{aligned}$$

  where $\eta = \frac{|\mathbf{P}_s|}{|\mathbf{P}_s|+|\mathbf{N}_s|}$ is the relative ratio of the number of positive edges and that of negative edges, and $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if a given predicate is true, or 0 otherwise. GAUC will be 1.0 for the perfect ranking list and 0.5 for a random ranking list [89].

- AUC (Area Under the Curve): AUC of the personalized ranking scores $\mathbf{r}$ w.r.t.

seed node s in signed networks is defined as follows [89]:

$$\text{AUC}_s = \frac{1}{|\mathbf{P}_s||\mathbf{N}_s|} \sum_{p \in \mathbf{P}_s} \sum_{n \in \mathbf{N}_s} \mathbb{I}(\mathbf{r}_p > \mathbf{r}_n)$$

where $\mathbf{P}_s$ is the set of positive nodes potentially connected by s, and $\mathbf{N}_s$ is the set of negative nodes. $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if a given predicate is true, or 0 otherwise. With an ideal ranking list, AUC should be 1 representing each positive sample is ranked higher than all the negative samples. For a random ranking, AUC will be 0.5. However, AUC is not a satisfactory metric for the link prediction task in signed networks because AUC is designed for two classes (positive and negative) while the link prediction in signed networks should consider three classes (positive, unknown, and negative) as described in the above.

A.4.2 Troll Identification

Suppose that we have a personalized ranking $\mathcal{R}$ in the ascending order of the trustworthiness scores w.r.t. a seed node (i.e., a node with a low score is ranked high) to have the same effect of searching trolls in the bottom of the original ranking in the descending order of those scores.

- MAP@k (Mean Average Precision): MAP@k is the mean of average precisions, AP@k, for multiple queries. Suppose that there are l trolls to be captured. Then, AP@k is defined as follows:

$$\text{AP@}k = \frac{1}{\min(l,k)} \left(\sum_{t \in \mathbf{T}} \text{Precision@}t \right)$$

where Precision@t is the precision at the cut-off t. Note that $\mathbf{T} = \{t | \mathbb{I}(\mathcal{R}[t]) =$

1 for $1 \leq t \leq k\}$ where $\mathcal{R}[t]$ denotes the user ranked at position t in the ranking $\mathcal{R}$, and $\mathbb{I}(\mathcal{R}[t])$ is 1 if $\mathcal{R}[t]$ is a troll. For N queries, MAP@k is defined as follows:

$$\text{MAP@}k = \frac{1}{N}\left(\sum_{i=1}^{N} \text{AP@}k\right)$$

- NDCG@k (Normalized Discount Cumulative Gain): NDCG is the normalized value of Discount Cumulative Gain (DCG), which is defined as follows:

$$\text{DCG@}k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{log_2(i)}, \text{ and } \text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}$$

  where rel_i is the user-graded relevance score for the i-th ranked item. Then, NDCG@k is obtained by normalizing using Ideal DCG(IDCG) which is the DCG for the ideal order of ranking.

- Precision@k and Recall@k: Precision@k (Recall@k) is the precision (recall) at the cut-off k in a ranking. Precision@k is the ratio of identified trolls in top-k ranking, and Recall@k is the ratio of identified trolls in the total trolls.

- MRR (Mean Reciprocal Rank): MRR@k is the mean of the reciprocal rank (RR) for each the top-k query response. RR is the multiplicative inverse of the rank of the first correct answer. Hence, for N multiple queries, MRR@k is defined as follows:

$$\text{MRR@}k = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{rank_i}$$

  where $rank_i$ is the rank position of the first relevant item in the top-k ranking. If there is no relevant item in the ranking for the i-th query, the inverse of the rank, ${rank_i}^{-1}$, becomes zero.

A.5 Discussion on Relative Trustworthiness of SRWR

In Section 4.3.1, we define the relative trustworthiness $\mathbf{r} = \mathbf{r}^+ - \mathbf{r}^-$ where $\mathbf{r}^+$ is for positive SRWR scores, and $\mathbf{r}^-$ is for negative SRWR ones. We show that $\mathbf{r}^+$ and $\mathbf{r}^-$ are *measures*, and $\mathbf{r}$ is a *signed measure* using definitions from measure theory [111]. We first introduce the definition of *measure* as follows:

Definition A.1 (**Measure** [111]). *A measure μ on a (finite) set Ω with σ-algebra $\mathcal{A}$ is a function $\mu : \mathcal{A} \to \mathbb{R}_{\geq 0}$ such that*

1. *(Non-negativity) $\mu(E) \geq 0 \; \forall E \in \mathcal{A}$,*
2. *(Null empty set) $\mu(\emptyset) = 0$,*
3. *(Countable additivity) $\mu(\bigcup_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} E_i$ for any sequence of pairwise disjoint sets, $E_1, E_2, \cdots \in \mathcal{A}$*

where σ-algebra $\mathcal{A}$ on Ω is a collection $\mathcal{A} \subseteq 2^{\Omega}$ s.t. it is nonempty, and closed under complements (i.e., $E \in \mathcal{A} \Rightarrow E^c \in \mathcal{A}$) and countable unions (i.e., $E_1, E_2, \cdots \in \mathcal{A} \Rightarrow \bigcup_{i=1}^{\infty} E_i \in \mathcal{A}$). The pair of $(\Omega, \mathcal{A})$ is called measurable space. ■

In probability theory, σ-algebra $\mathcal{A}$ describes all possible events to be measured as probability. Note that $\mathbf{r}^+$ and $\mathbf{r}^-$ are joint probabilities of nodes and signs, i.e., $\mathbf{r}^+_u = P(N = u, S = +)$ and $\mathbf{r}^-_u = P(N = u, S = -)$ where N is a random variable of nodes, and S is a random variable of the surfer's sign. Note that N takes an item from σ-algebra $\mathcal{A}$. The following property shows that $\mathbf{r}^+$ and $\mathbf{r}^-$ are (non-negative) measures.

Property 5. *Suppose Ω is the set $\mathbf{V}$ of nodes, and σ-algebra $\mathcal{A}$ on Ω is 2^{Ω}. Let $\mu^+ = P(N, S = +)$ and $\mu^- = P(N, S = -)$. Then, both μ^+ and μ^- are (non-negative) measures according to Definition A.1.*

Proof. For any $E \in \mathcal{A}$, $\mu^+(E) \geq 0$ and $\mu^+(\emptyset) = 0$ are obviously true since $P(N, S = +)$ is a probability; hence, $P(E, S = +) \geq 0$ and $P(\emptyset, S = +) = 0$. Let $(E_n)_{n\in\mathbb{N}}$ be a sequence of pairwise disjoint sets where $E_n \in \mathcal{A}$. Since the sets in the sequence are mutually disjoint, the following holds:

$$P\left(\bigcup_{n\in\mathbb{N}} E_n, S = +\right) = \sum_{n\in\mathbb{N}} P(E_n, S = +)$$

Therefore, $\mu^+ = P(N, S = +)$ is a measure by Definition A.1. Similarly, $\mu^- = P(N, S = -)$ is also a measure. □

Next, we introduce the definition of *signed measure*, a generalized version of measure by allowing it to have negative values.

Definition A.2 (Signed Measure [111]**).** *Given a set Ω and σ-algebra $\mathcal{A}$, a signed measure on $(\Omega, \mathcal{A})$ is a function $\mu : \mathcal{A} \to \mathbb{R}$ such that*

1. *(Real value) $\mu(E)$ takes a real value in $\mathbb{R}$,*
2. *(Null empty set) $\mu(\emptyset) = 0$,*
3. *(Countable additivity) $\mu(\bigcup_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} E_i$ for any sequence of pairwise disjoint sets, $E_1, E_2, \cdots \in \mathcal{A}$* ■

Note that *electric charge* is a representative example of signed measure. Then, the following lemma indicates the difference between two non-negative measures is a signed measure.

Lemma A.2 (Difference Between Two Non-negative Measures [111]**).** *Suppose we are given non-negative measure μ^+ and μ^- on the same measurable space $(\Omega, \mathcal{A})$. Then, $\mu = \mu^+ - \mu^-$ is a signed measure.*

Proof. Since μ^+ and μ^- are non-negative, μ is located between $-\infty$ and ∞. Also, $\mu(\emptyset) = \mu^+(\emptyset) - \mu^-(\emptyset) = 0$. Moreover, μ is countable additive, i.e.,

$$\mu\left(\bigcup_{i=1}^{\infty} E_i\right) = \mu^+\left(\bigcup_{i=1}^{\infty} E_i\right) - \mu^-\left(\bigcup_{i=1}^{\infty} E_i\right) = \sum_{i=1}^{\infty}\left(\mu^+(E_i) - \mu^-(E_i)\right) = \sum_{i=1}^{\infty}\mu(E_i)$$

Hence, $\mu = \mu^+ - \mu^-$ is a signed measure according to Definition A.2. □

Lemma A.2 implies that the relative trustworthiness $\mathbf{r} = \mathbf{r}^+ - \mathbf{r}^-$ is a signed measure. The trustworthiness $\mathbf{r}_u$ measures a degree of trustworthiness between seed node s and node u: if $\mathbf{r}_u > 0$, seed node s is likely to trust node u as much as $\mathbf{r}_u$ while if $\mathbf{r}_u < 0$, s is likely to distrust u as much as $\mathbf{r}_u$.

요 약

다양한 실세계 자연 현상에서의 관계들은 소셜 네트워크, 하이퍼링크 네트워크와 단백질 상호작용 네트워크와 같이 정점과 간서의 그래프로 표현된다. 이러한 네트워크를 분석하는 것은 실세계의 현상을 이해하는데 매우 중요하다. 다양한 그래프 분석 기법중에 랜덤 워크라는 기법이 만족스러운 성능과 함께 많은 그래프 마이닝 응용에 널리 활용되어 왔다. 그러나 대다수의 실세계 그래프는 그 규모가 굉장히 크고 다양한 라벨 정보와 함께 복잡하게 표현된다. 전통적인 랜덤 워크 기반의 기법들은 계산량이 많이 요구되고, 랜덤 워크를 하는데 있어서 다양한 라벨 정보를 전혀 고려하지 않아 라벨로 표현되는 그래프의 고유한 특성이 무시되게 된다. 그래서 이와 같이 복잡하면서 대규모 그래프에서는 랜덤 워크의 실질적 활용이 제한되어왔다.

본 학위 논문에서는 랜덤 워크 기반의 대규모 실세계 그래프 분석의 기술적 한계를 해결하고자 한다. 실세계 그래프는 고유한 구조적 특징들을 가지고 있으며 이러한 구조적 특징들은 속도와 품질의 측면에서 랜덤 워크의 성능을 향상시키는데 기반이 될 수 있다. 이러한 아이디어를 활용하여, 대규모의 라벨이 없는 일반적인 네트워크에서 랜덤 워크 기반의 개인화된 정점 랭킹 계산을 빠르고, 확장성 있고 정확하게 구하는 기법을 제안한다. 또한 부호화된 네트워크 또는 지식 베이스와 같은 라벨이 있는 그래프에서 개인화된 정점 랭킹과 관계 추론을 위한 랜덤 워크 기반의 모델을 제안한다.

다양한 실세계 그래프에서 광범위한 실험을 통해 본 학위 논문에 의해 제안된 방법과 모델의 효과성을 보인다. 제안하는 방법은 다른 경쟁 기법들과 비교했을 때 최대 100배 더 큰 그래프를 처리할 수 있고, 최대 130배 적게 메모리를 사용하면서, 최대 9배 빠른 속도를 보이며, 결과적으로 수 십억 규모의 그래프에서 랜덤 워크 기반의 개인화된 정점 랭킹을 성공적으로 구할 수 있다. 또한, 제안하는 랜덤 워크 기반의

모델들은 부호화된 네트워크와 지식 베이스와 같은 라벨이 있는 그래프에서 부호 예측, 간선 예측, 이상 현상 탐지, 관계 추론 등의 다양한 응용에서 다른 경쟁 모델들보다 더 좋은 예측 성능을 보인다.

주요어 : 그래프 마이닝, 그래프 랜덤 워크, 랜덤 워크와 재시작 모델, 실세계 그래프 특징, 대규모 그래프, 부호화된 네트워크, 간선 라벨이 있는 그래프
학번 : 2015-31053