IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

A Survey on Personalized PageRank Computation Algorithms

Sungchan Park[1], Wonseok Lee[1], Byeongseo Choe[1], and Sang-goo Lee[1]

[1] Seoul National University, 138-dong 418-ho, 1 Gwanak-ro, Gwanak-gu, Seoul, Republic of Korea

Corresponding author: Sungchan Park (e-mail: baksalchan@ europa.snu.ac.kr).

ABSTRACT Personalized PageRank (PPR) is an important variation of PageRank, which is a widely applied popularity measure for Web search. Unlike the original PageRank, PPR is a node proximity measure that represents the degree of closeness among multiple nodes within a graph. It is also widely applied to diverse domains, such as information retrieval, recommendations, and knowledge discovery, due to its theoretical simplicity and flexibility. However, computing PPR in large graphs using naïve algorithms such as iterative matrix multiplication and matrix inversion is not fast enough for many of these applications. Therefore, devising efficient PPR algorithms has been one of the most important subjects in large-scale graph processing. In this paper, we review the algorithms for efficient PPR computations, organizing them into five categories based on their core ideas. Along with detailed explanations including recent advances and their applications, we provide a multifaceted comparison of the algorithms.

INDEX TERMS Personalized PageRank, Graph Data Mining, Graph Analysis

I. INTRODUCTION

Finding the proximity among multiple entities using distance/similarity measures is one of the core operations of data mining and knowledge discovery. Likewise, finding the closeness among multiple nodes within graphs is also an important problem in graph data mining. *Personalized PageRank* (also known as "*Random Walk with Restart*") is one of most intensely studied node proximity measures for graph data mining and has also been adopted by a wide range of applications.

Personalized PageRank (PPR) is a variation of PageRank, which is a way of measuring the importance of hyperlinked webpages [32]. The core idea of PageRank is the introduction of the random walk model. It assumes that a walker resides on a node at a specific time and travels on the graph through its edges. PageRank of each node can be seen as the probability that the walker resides on each node when it infinitely "random walks" or jumps to a random node with a constant probability. Thus, PageRank can measure the importance of each node considering the link structure of graphs and it has been successfully adopted by Web search systems such as Google. Like the original PageRank, PPR is also defined using a random walk model. However, PPR assumes that a traveling walker infinitely returns (jumps) to their "*restarting nodes*", which is a specific set of nodes, instead of all nodes. In PPR, the result is skewed toward the restarting nodes, and thus, PPR is a measure of the proximity of each node to the restarting nodes. A node with a high PPR score can be considered as a node that is close to the restarting nodes. It produces relatedness scores among nodes such as the traditional distance/similarity measures, i.e., the shortest path distance and the maximal flow. PPR considers every possible direct/indirect connection among nodes while the traditional measures utilize only limited information. For example, the shortest path distance considers only the shortest connection between two nodes, while other connection information is not used. In contrast, PPR considers every path for reaching the target node that a "random walker" can follow. Thus, PPR can reflect the overall structural features of graphs.

Due to its merits, PPR has been applied to a wide range of applications such as information retrieval, context-aware recommendations, social network analysis, computational linguistics, image processing, anomaly detection, and bioinformatics.

Since PPR can be interpreted as a simple linear equation, it can be computed using basic equation solving algorithms such as iterative matrix multiplication and matrix inversion. However, for large graphs consisting of over 10 million nodes, such as social networks and the World Wide Web, these basic methods are not fast enough to meet the requirements of most applications. Therefore, finding efficient algorithms for PPR has been one of the major

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2952653, IEEE Access

subjects of graph data processing research and, consequently, many advanced algorithms have been proposed.

The advanced algorithms for the computation of PPR have a wide range of diverse and distinctive features. For example, sampling-based methods return answers quickly while matrix inversion requires heavy computational costs; however, sampling-based methods do not guarantee error bounds while matrix inversion always returns exact results. To properly utilize PPR, one should choose the most appropriate method among multiple algorithms while considering their characteristics. However, to the best of our knowledge, there is yet to be a comprehensive survey on PPR computations with a multifaceted comparison of the algorithms.

In this paper, we summarize PPR computational algorithms. We group them into five categories: optimized iterative equation solving, optimized direct equation solving, bookmark coloring algorithms, dynamic programming, and Monte-Carlo sampling. Our survey includes an overview of the basic ideas and recent advancements of each approach, and compares the approaches/algorithms according to the following criteria.

- **Precomputation Time**: Most recent algorithms require a precomputation phase to optimize query answering time. Though precomputation is not added to the query response time, it should be optimized so as to keep the graph update overhead in control. If the query results should be updated daily, the precomputation time should not be longer than a day.
- **Query Answering Time**: The query results should be returned as quickly as possible. The query answering time largely affects the overall running time of data mining tasks that require frequent PPR computations (e.g., node clustering).
- **Auxiliary Data Size**: Algorithms that require precomputation phase often construct auxiliary data structures during the phase (e.g., sampled random walk traces). If the size of the data is too large to be contained in main memory, overall system performance including query answering time can be severely degraded. Thus, reducing the size of the auxiliary data without degrading other criteria is one of the major challenges.
- **Precision**: The quality of the output varies with the computation methods. If a certain degree of error is not tolerable, algorithms that cannot meet the requirements cannot be adopted regardless of its performance in other features.

We expect that our study will provide effective guidance for choosing proper algorithms for PPR computations and conducting further research on the subject.

The remainder of this paper is organized as follows. Section 2 presents the theoretical basis of PPR. Several important and useful characteristics of PPR are also introduced. Section 3 presents the use cases of PPR. Section 4 presents the advanced PPR algorithms categorized into five types of approaches. In section 5, detailed discussions with a multifaceted comparison of the approaches are provided. Section 6 concludes the report by summarizing the survey.

II. PRELIMINARIES: RANDOM WALK, PAGERANK, AND PERSONALIZED PAGERANK

In this section, we describe theoretical basis of PPR, including its definition and characteristics. For self-contained explanation, we starts with random walk and PageRank that PPR is based on, then we provide detailed discussions on PPR based on the previously explained concepts. The difference between PageRank and PPR is also explained. Table 1 provides the definitions of the symbols that are used in the following sections.

TABLE Ⅰ Table of symbols.

Symbol	Definition
G	Given graph
N	Number of nodes in G
P	Transition matrix of G
c	Damping factor ($1-c$ = restart probability)
$PPR(\vec{v})$	PPR vector of a given query vector $\vec{v}$
$PPR_0(\vec{v})$	Initial guess vector for a given query vector $\vec{v}$
n_i	The i^{th} node
$\overrightarrow{v_i}$	A unit vector defined as $\overrightarrow{v_i}[k] = \begin{cases} 1, & k = i \\ 0, & otherwise \end{cases}$
$O(n_i)$	Out neighbor node set of n_i
$deg(n_i)$	Degree of n_i

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2952653, IEEE Access

A. Basics on Random Walk

PPR is one random walk-based proximity measure. Thus, we first start with a brief explanation of the random walk in order to provide a self-contained introduction to PPR. The *random walk* model assumes that a particle, which is called a *random walker*, walks on a given graph through its edges. For example, we can think that a random walker that resides on node n_1 will be on n_2 or n_3 from the above assumption (Fig, 1).

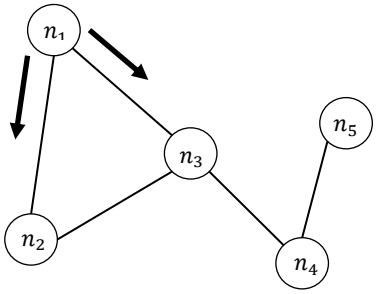


FIGURE 1. Random walk example

The state of a random walker can be expressed as a probability vector $\vec{v}$ with N dimensions. Each dimension of $\vec{v}$ corresponds to each node, and the entry represents the probability that a random walker resides on the node. For example, a state vector $\overrightarrow{v_{(0)}} = (1, 0, 0, 0, 0)^T$ represents the state that a random walker resides on node n_1 without uncertainty, and a state vector $\overrightarrow{v_{(1)}} = (0, 1/2, 1/2, 0, 0)^T$ represents the state that a random walker may reside on node n_2 or n_3 with an even probability. We can say that if the current state is $\overrightarrow{v_{(0)}}$, the next state is $\overrightarrow{v_{(1)}}$ from the assumption of the random walk. Generally, the transition of a random walker can be represented as a probability matrix P^T, which is called the *transition matrix*. The entry of P^T can be defined by the following rule:

$$P^T{}_{ij} = \begin{cases} \dfrac{1}{|O(n_i)|}, & j \in O(n_i) \\ 0, & otherwise \end{cases} \quad (1)$$

The transition matrix P^T for the graph in Fig. 1 can be calculated as follows:

$$P^T = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1/3 & 1/3 & 0 & 1/3 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad (2)$$

We can use the matrix P^T to compute the state vector of the next step from the state vector of the current step using the following equation: $\overrightarrow{v_{(k+1)}} = P\overrightarrow{v_{(k)}}$. Therefore, the state of a random walker after k[th] steps can be expressed as $\overrightarrow{v_{(k)}} = P^k\overrightarrow{v_{(0)}}$, where $\overrightarrow{v_{(0)}}$ denotes the initial state.

B. Definition of PageRank

Before discussing PPR, we need to review PageRank since PPR is a generalization of PageRank, a popularity measure for webpages.

PageRank assumes an imaginary Web surfer that visits one webpage at a time and randomly moves through hyperlinks. With the assumption, PageRank measures the popularity of each node by the probability of the surfer resides on each node after infinite number of steps.

This assumption can be precisely interpreted into the random walker model. Thus, PageRank can be represented by the probability distribution of the random walker over the graph.

In random walker model, the probability vector converges after infinite steps. We can compute the converged probability vector by solving the following equation:

$$\vec{r} = P\vec{r}$$

P presents the $N \times N$ transition matrix of the Markov chain that is derived from the graph. The i[th] entry of vector $\vec{r}$ represents the probability that the random walker resides at node i.

The probability is changed by introducing the damping factor. The justification within the random walker model is that the walker does not move over an infinite number of links but gets bored sometimes and jumps to another node at random. The following equation represents PageRank with the damping factor:

$$\vec{r} = cP\vec{r} + \frac{1}{N}(1-c)\vec{1} \quad (3)$$

Thus, $P\vec{r}$ represents the transition from the current state, and $\frac{1}{N}\vec{1}$ represents the uniform state. In summary, equation (3) represents the recursive process through which a random walker follows the Markov process with probability c and jumps to random node with probability $1-c$. The solution $\vec{r}$ of equation (3) is PageRank vector, and PageRank value of a specific node i is the i[th] entry of $\vec{r}$.

PageRank value represents the **centrality** of node i within the graph. The value is larger when the node is related to other important nodes. The value is smaller when the node has few edges or PageRanks of the related nodes are also small. The sum of the value is 1 because it is sum of the probabilities.

C. Definition of Personalized PageRank

PPR can also be represented in recursive form. The following equation represents PPR:

$$\vec{r} = cP\vec{r} + (1-c)\vec{s} \quad (4)$$

The only difference from the original PageRank is the second term where $\frac{1}{N}\vec{1}$ is replaced with $\vec{s}$. It means that now the random walker jumps to the nodes that are specified by the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2952653, IEEE Access

probability vector $\vec{s}$. For example, if the i^{th} entry of $\vec{s}$ is 1 and all other entries are zeros, the random walker continuously returns to node i with probability $1-c$. In PPR, the probability vector $\vec{r}$ is relatively more skewed toward $\vec{s}$ than in PageRank. The solution $\vec{r}$ of the equation (4) is PPR vector, which is denoted as $PPR(\vec{s})$. The j^{th} entry of $PPR(\vec{s})$ represents the **proximity** of node j from the nodes that are specified by $\vec{s}$. If the i^{th} entry of $\vec{s}$ is 1 and all other entries are zeros, $PPR(\vec{s})$ represents the proximity of each node from node i. If $PPR(\vec{s})[j]$ is higher than $PPR(\vec{s})[k]$, we can say that node j is more strongly connected to node i than node k is.

In summary, the difference between PageRank and PPR is that PPR assumes that the random walker randomly returns to specific states (i.e., query states), which is unlike PageRank that assumes that the random walker returns to any node with uniform probability.

The *Personalized PageRank problem* is defined as the problem of computing $PPR(\vec{s})$ vector when the graph and the restart vector $\vec{s}$ are given. The *fully Personalized PageRank problem*, which is also a well-studied problem, is defined as the problem of computing the $PPR(\overrightarrow{v_i})$ vector for every node n_i in the given graph.

The thing that makes the Personalized PageRank problem harder than PageRank problem is that PPR has infinite possible queries. The PageRank vector remains unchanged unless the graph is changed. Thus the vector can be computed off-line and reused until the graph is updated. However, $PPR(\vec{s})$ vectors should be recomputed when node proximity value is needed since $\vec{s}$ can be any stochastic vector. Considering that the tasks require PPR computation such as Web search and graph data mining need frequent proximity computation, it is obvious that the algorithms for PageRank computation are not sufficient for PPR computation. Consequently, there exist tons of studies focusing on efficient PPR computation. We will provide detailed discussion on them in section 4.

D. Characteristics of Personalized PageRank

PPR has some useful characteristics. The most important and frequently utilized ones are the *linearity* and the *decomposition theorem* [23].

The following equation represents the *linearity*:

$$PPR(w_1\vec{u} + w_2\vec{v}) = w_1 PPR(\vec{u}) + w_2 PPR(\vec{v}) \qquad (5)$$

It means that we can compute $PPR(w_1\vec{u} + w_2\vec{v})$ when we know $PPR(\vec{u})$ and $PPR(\vec{v})$ without solving a linear equation. For example, if $PPR((0,0,0,1,0)^T)$ and $PPR((1,0,0,0,0)^T)$ are known, we can compute PPR vector of any linear combination of $(0,0,0,1,0)^T$ and $(1,0,0,0,0)^T$ such as $(0.2,0,0,0.8,0)^T$ via weighted summation of the known PPR vectors. This property is utilized by several optimization methods such as Bookmark Coloring Algorithm (Section 4. C).

The *decomposition theorem* is also an interesting property that is closely related to the graph structure. The following equation represents the decomposition theorem:

$$PPR(\overrightarrow{v_1}) = c\overrightarrow{v_1} + \frac{1-c}{|O(n_1)|}\sum_{n_j \in Out(n_1)} PPR(\overrightarrow{v_j}) \qquad (6)$$

It means that we can compute $PPR(\overrightarrow{v_1})$ if we know all PPR vectors from the out neighbor nodes of n_1. Assume that we want to compute $PPR(\overrightarrow{v_1})$, and we know PPR of all out neighbor nodes. Then, in the example graph in Fig. 1, the following equation holds according to the decomposition theorem:

$$PPR(\overrightarrow{v_1}) = c\overrightarrow{v_1} + \frac{1-c}{2}PPR(\overrightarrow{v_2}) + \frac{1-c}{2}PPR(\overrightarrow{v_3}) \qquad (7)$$

Therefore, we can compute $PPR(\overrightarrow{v_1})$ as the summation of vectors if we know the vectors corresponding to $PPR(\overrightarrow{v_2})$, $PPR(\overrightarrow{v_3})$, $PPR(\overrightarrow{v_4})$ and $PPR(\overrightarrow{v_5})$ beforehand. Since solving linear equation is an expensive task, utilizing the above properties is one of major approaches for PPR computations.

Another important property of PPR is that it can be rewritten in summation form as follows:

$$PPR(\vec{s}) = (1-c)\sum_{k=0}^{\infty} c^k P^k \vec{s} \qquad (8)$$

This interpretation of PPR can be seen as iterative summation of the state vectors for each step of a random walker that follows the transition behavior of PPR. If we assume that the initial state $\vec{r}^{(0)} = \vec{s}$, then the state vector of the 1st step $\vec{r}^{(1)}$ can be computed as $(1-c)\vec{s} + cP\vec{s}$ by equation (4). The state vector of the 2nd step can be obtained in the same fashion as follows: $\vec{r}^{(2)} = (1-c)\vec{s} + cP\vec{r}^{(1)} = (1-c)\vec{s} + (1-c)cP\vec{s} + c^2P^2\vec{s}$. If we repeat the former process infinitely, we can eventually reach the equation $\vec{r}^{(\infty)} = (1-c)\vec{s} + (1-c)cP\vec{s} + (1-c)c^2P^2\vec{s} + (1-c)c^3P^3\vec{s} + \cdots$, and the equation can be rewritten as the summation form of the equation (8). This property is utilized in several important methods including sampling-based algorithms [6][22].

Another property is *weighted symmetry*. Basically, PPR is not a symmetric measure unlike other widely used similarity measures such as Euclidean distance. That is, we cannot say that the proximity of node i from node j is equal to that of node j from node i in terms of PPR. In other words, we cannot guarantee that $PPR(\overrightarrow{v_i})[j] = PPR(\overrightarrow{v_j})[i]$ always holds. However, PPR also has the property of partial symmetry when the given graph is an undirected graph. Equation (9) shows the property [25]:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2952653, IEEE Access

$$\deg(i) \cdot PPR(\overrightarrow{v_i})[j] = \deg(j) \cdot PPR(\overrightarrow{v_j})[i] \quad (9)$$

Though it is not a perfect symmetric property, it can be utilized for optimization by avoiding repetitive computations. If we have the vector $PPR(\overrightarrow{v_j})$ beforehand, we can get the $PPR(\overrightarrow{v_k})[j]$ for any node k by using the property.

PPR also can be viewed as a solution of the linear system $(I - cP)\vec{r} = (1 - c)\vec{s}$. The system matrix $(I - cP)$ of the above linear system has the following properties [31]:

1. $(I - cP)$ is an M-matrix.
2. $(I - cP)$ is nonsingular.
3. The row sums of $(I - cP)$ are $1 - c$.
4. $\|I - cP\|_\infty = 1 + c$.
5. Since $(I - cP)$ is an M-matrix, $(I - cP)^{-1} \geq 0$.
6. The row sums of $(I - cP)^{-1}$ are $(I - c)^{-1}$. Therefore, $\|(I - cP)^{-1}\|_\infty = (I - c)^{-1}$.
7. Thus, the condition number $\kappa_\infty(I - cP) = (1 + c)/(1 - c)$

The above properties guarantee the *existence* and *uniqueness* of the solution of the given linear system. They also assure that the solution can be computed using iterative methods for solving linear equations such as the Jacobi method.

III. APPLICATIONS

In this section, we overview the applications of PPR. As a well-studied node proximity measure, PPR has been applied to a broad range of fields including information retrieval, item recommendations, social network analysis, computational linguistics, computer vision, bioinformatics, probabilistic reasoning, etc.

A. Personalized Web Search

One of well-known application examples of PPR is personalized web searches. Applying PPR to personalized web searches was suggested at the birth of PageRank [32]. Its actual application was studied later by Haveliwala *et al.* [7]. In this paper, the restart vector $\vec{s}$ is defined by topics that represent personal preferences. In particular, let T_j be the set webpages in the category c_j on the Open Directory Project[1]. When computing PageRank for topic c_j, the non-uniform vector $\overrightarrow{v_j}$ is used in place of the uniform vector $\frac{1}{N}\vec{1}$:

$$v_j[i] = \begin{cases} \frac{1}{|T_j|}, & i \in T_j \\ 0, & i \in T_j \end{cases} \quad (10)$$

By using the biased restart vector $\overrightarrow{v_j}$, the result of PageRank can represent the importance of each page in terms of a certain topic. The result of the experiment shows that using a biased restart vector in place of an unbiased vector significantly improves the precision of an information retrieval system.

Gleich *et al.* [43] also proposed an approximate PPR algorithm and personalized web search system that can provide personalized search results to users without violating the users' privacy. Dou *et al.* [51] presented a large-scale evaluation framework for personalized search strategies and analyzed the effects of many personalization strategies.

B. Social Network Analysis

PPR is used for measuring importance of each users in SNSs to analyze social network structures. Garcia *et al.* [55] analytically characterized all the possible values of PPR for any node, and introduced a new concept concerning the competitivity and leadership in networks. Pedroche *et al.* [14] introduced a new parameter, the frequency, to the Leadership group, which is a group of nodes that have higher PageRanks than others. Then, they analyzed some graphs using the Leadership group while controlling the biasing factor ϵ. Additionally, PPR has been used as a link prediction tool of SNSs [59]. Backstrom *et al.* [60] developed an algorithm called the "Supervised Random Walk" that combines the information of a network structure and edge attributes to predict the links in social network. They used the node and edge attributes to guide the random walks toward the target node. Liu *et al.* [61] proposed a link prediction method based on the local random walk that has much lower computational complexity. Xia *et al.* [44] showed that Random Walk with Restart (RWR) can be utilized to determine the relevancy in a birelational network in the bibliographic domain. Tong *et al.* [45] introduced the "center-piece subgraphs" problem of a social network and proposed a fast subgraph extraction algorithm. Jung *et al.* [62] proposed Signed Random Walk with Restart (SRWR) for personalized rankings in signed networks using a signed surfer. Devooght *et al.* [63] introduced a random walk based modularity measure, which is computed using the paths instead of the traditionally used edges, for analyzing social networks.

C. Computational Linguistics

PPR has made some achievements in the field of Computational linguistics. Liu *et al.* [16] applied PPR to generate query-based multidocument summarizations. They used PPR to rank the personalized prior probability of each sentence, which is computed using their salience model and relevance model. Agirre *et al.* [15] used PPR to solve the graph-based word sense disambiguation (WSD) problem. In WSD, a graph consists of nodes, which represent word senses, and edges, which represent relations between pairs of word senses. PPR is utilized when performing disambiguation by applying a ranking algorithm to a graph. Pershina *et al.* [64] introduced PPR-based random walk method to solve Named Entity Disambiguation (NED) problem. They used PPR algorithm on a graph where the

[1] Open Directory Project, http://odp.org/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2952653, IEEE Access

vertices represent candidate links and the edges represent links in Wikipedia. They achieved state-of-the-art performance on a 27.8K named entity mention dataset.

D. Computer vision

Computer vision is one of the fields where PPR has also been widely applied. Kim *et al.* [19] addressed a multilabel supervised image segmentation problem when initial labels of some pixels are given. They introduced the generative model for image segmentation using the steady-state probability of RWR.

Since RWR considers all relevance relations between the nodes in a graph (image), it is effective at addressing the texture problem. Ham *et al.* [20] proposed a generalized random walk with restart (GRWR), which is a generalized version of RWR that adopts local and nonlocal approaches for image regularization. They applied GRWR to depth map upsampling and interactive image segmentation and showed that the GRWR is more robust to outliers and can aggregate texture information better. Wang *et al.* [65] utilized PPR to refine image annotations. After a relevance model-based algorithm determines the candidate annotations, RWR is used to rerank the annotations based on the corpus information and original confidence. Kim *et al.* [66] proposed a multiscale saliency detection algorithm that uses RWR to refine a saliency map. Similarly, Kim *et al.* [67] proposed a spatiotemporal saliency detection algorithm for video sequences based on RWR. Lee *et al.* [68] proposed a robust dense stereo reconstruction algorithm using RWR. Kim *et al.* [69] devised a modified data-driven RWR framework that can incorporate locally adaptive and data-driven restarting probabilities to handle the colorization problem of grayscale images. Oh *et al.* [70] presented a probabilistic method for correspondence matching using RWR.

E. Bioinformatics

PPR can also be applied to scientific data analysis [21][47][48]. Iván *et al.* analyzed protein interaction networks using PPR [21]. They applied PPR to analyze protein-protein interaction (PPI) networks that connect interacting proteins. Sun *et al.* [71] proposed a global network-based computational framework, which was called RWRlncD, to infer potential human lncRNA-disease associations by implementing the method on an lncRNA functional similarity network. Chen *et al.* [72] proposed an Improved Random Walk with Restart for a lncRNA-Disease Association Prediction (IRWRLDA) model to predict novel lncRNA-disease associations by incorporating lncRNA expression similarity and disease semantic similarity. Chen *et al.* [73] proposed a Network-based Random Walk with Restart on a Heterogeneous network (NRWRH) to predict potential drug-target interactions on a large scale under the hypothesis that similar drugs often target similar target proteins. Li *et al.* [74] identified novel epigenetic factors by using a computational method that applied RWR algorithm on a protein-protein interaction (PPI) network using reported epigenetic factors as seed nodes. Blatti *et al.* [75] presented a network-based method, which involves RWR, for ranking the genes or properties related to a given gene set. Chipman *et al.* [76] presented a method based on RWR, which captures aspects of the network topology to classify potential genetic interactions and applied it to biological networks.

F. Others

There are many other fields where PPR is applied. The FolkRank is a folksonomy-based algorithm that is used for tag recommendations. Kim *et al.*[17] proposed a new way to efficiently compute the FolkRank by representing it as a linear combination of PPR vectors.

In first-order probabilistic representation systems, inference by grounding can be very computationally expensive. Wang *et al.* [42] proposed a first-order probabilistic language to approximate the local grounding by applying PPR to a small graph.

Nykl *et al.* [77] evaluated a citation network that was built using the ISI Web of Science database. Their aim was to find an evaluation method that best matches the list of authors who received ACM Fellowships or ACM SIGs. The best ranking method included PPR where the personalization is based on PageRank journal values.

Local graph diffusion is an effective tool for solving graph clustering problems. Avron *et al.* [78] proposed an efficient local algorithm for approximating a graph diffusion, which generalizes PPR and the heat kernel.

Tabrizi *et al.* [52] proposed a Personalized PageRank Clustering (PPC) algorithm that utilizes the random walk and modularity to accurately reveal the inherent clusters of graphs. It also gives a hierarchy of the clusters given linear time and space complexity.

Andersen *et al.* [50] presented a generalized local partitioning algorithm for undirected graphs to strongly connect directed graphs by computing PPR vector.

Guo *et al.* [13] proposed the Access Time-length and Frequency-based PageRank to prefetch web pages for web page caching.

With respect to databases, Balmin *et al.* [38] devised a method called "ObjectRank", which is a variation of PPR, to perform keyword searches based on authority. Chakrabarti [40] proposed HubRank, which is a proximity search platform for Entity-Relation graphs, using the dynamic PPR.

IV. COMPUTATION OF PERSONALIZED PAGERANK

In this section, we introduce the two basic algorithms for computing PPR—the power iteration and equation solving—and summarize the advanced studies about computing PPR.

There are two basic algorithms to solve the problem: the power iteration and direct solving. These algorithms are directly derived from the definition of PPR. The power iteration performs the following iteration until $\vec{r}^{(n)}$ converges: $\vec{r}^{(n+1)} = cP\vec{r}^{(n)} + (1-c)\vec{s}$. The direct solution method solves the following equation: $\vec{r} = cP\vec{r} + (1-c)\vec{s}$. Thus, the solution $\vec{r} = (1-c)(I-cP)^{-1}\vec{s}$ is the answer to PPR query about the query state $\vec{s}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2952653, IEEE Access

These algorithms have obvious drawbacks. First, the power iteration algorithm requires multiple matrix-vector multiplications. One iteration requires $O(N^2)$ time. The direct solution algorithm is even worse. As we can see in the equation, it includes the computation of the inverse matrix. Since its time complexity is $O(N^3)$, it can be applied to small sized graph data only. Furthermore, generally, matrix inversion does not preserve the sparsity of the original matrix. This means that we cannot utilize the sparse matrix representation of graphs.

From the observation, we can see that the basic algorithms cannot handle large-scale graphs. To solve the situation, there have been many studies on the efficient computation of PPR.

Though there have been many studies on the efficient computation of PPR, most of them can be categorized into one of the following categories.

- Optimized iterative equation solving
- Optimized direct equation solving
- Bookmark coloring algorithm
- Dynamic programming method
- Monte-Carlo sampling-based method

Each category has distinctive characteristics in terms of accuracy and costs. In the following sections, we provide detailed discussions of each category.

A. Optimized Iterative Equation Solving

These studies can be viewed as advanced versions of the power iteration. Actually, the power iteration itself is identical to the *Jacobi method*, which is the most basic iterative method of solving general linear equations. Since equation (4) is a linear equation, any iterative method for linear equation solving can be applied, and as we stated in section 2.D, PPR equation converges to the unique solution with iterative methods due to its mathematical properties. Therefore, linear equation solving algorithms such as the Gauss-Seidel method, Successive Overrelaxation (SOR), GMRES, and Multigrid methods can be applied to the problem. Though these iterative methods cannot produce exact solutions to the problems, their errors can be controlled and the error bound can be strictly defined unlike other approximation algorithms such as sampling-based methods. On the top of applying those general equation solvers, diverse techniques are devised for the specific problem of computing PPR.

One of the most recent studies in this category utilizes the GMRES [10]. They utilize the GMRES for solving the equation (4), and applied a *preconditioning* method to accelerate the convergence of iterative algorithm. General concept of preconditioning is solving $MA\vec{x} = M\vec{b}$ to solve $A\vec{x} = \vec{b}$. The matrix M is called the *preconditioner*, and a well-chosen preconditioner reduces the number of iterations. It is known that the matrices that are close to A^{-1} can be good preconditioners. To construct a good preconditioner, they apply core-tree decomposition. They use an inversion of the tree-like part of the original graph as a preconditioner. The following table shows the result of the performance evaluation.

Overall, their method performs better than the power iteration and naïve GMRES. They report that their algorithm reduces the iteration count to achieve the same accuracy by 1/5 comparing to power iteration and 1/3 to naïve GMRES. However, their algorithm still requires several minutes to process each query on large graphs with millions of nodes.

Additionally, there are several studies seeking to accelerate the iterative method by tuning the power iteration. The extrapolation method is an example of the approach [30]. They accelerate the power iteration by subtracting nonprincipal eigenvectors periodically. They report that the extrapolation method can reduce the number of iterations of the power iteration by 1/2. It is an effective methods to solve the problem, but it is developed mainly for PageRank problem and its scalability for PPR computation is not guaranteed.

B. Optimized Direct Equation Solving

The direct equation solving method that solves linear equations by computing the inversions is not generally recommended due to its high complexity. However, when approximated answers are acceptable, revised versions of the method that produce approximated solutions can be an alternative method to solve PPR problems.

One of pioneering studies in this category introduces the low rank approximation to reduce the inversion complexity [1]. They use singular value decomposition (SVD) for the low rank approximation. By using the Low rank approximation and Sherman-Morrison lemma [79], it computes PPR using a smaller inversion matrix. It can be achieved as follows. When $\tilde{P} = USV$ and $\tilde{\Lambda} = (S^{-1} - cVU)^{-1}$, then $(I - cP)^{-1} \cong \left(I - cU\tilde{\Lambda}V\right)$ holds according to the Sherman-Morrison lemma. Since the dimension of S is far smaller than P, $\left(I - cU\tilde{\Lambda}V\right)$ can be computed more efficiently than $(I - cP)^{-1}$. This method provides a good approximation for PPR problem; however, its error bound is hard to control. They reported that their algorithm performs better than direct inversion and the power iteration. However, they use a relatively small graph (with 315K nodes) in the experiment.

There is also a study that optimizes direct equation solving without introducing the approximation [9]. They reorder the dimensions of the overall equations before the inversion to preserve the sparsity of the original matrix.

Generally, matrix inversion can be computed via LU decomposition, and the result is computed as $U^{-1}L^{-1}$. With the proper reordering, the sparsity of the original matrix is also preserved with respect to U^{-1} and L^{-1}, and the sparse matrices can be stored and processed more efficiently than dense matrices.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2952653, IEEE Access

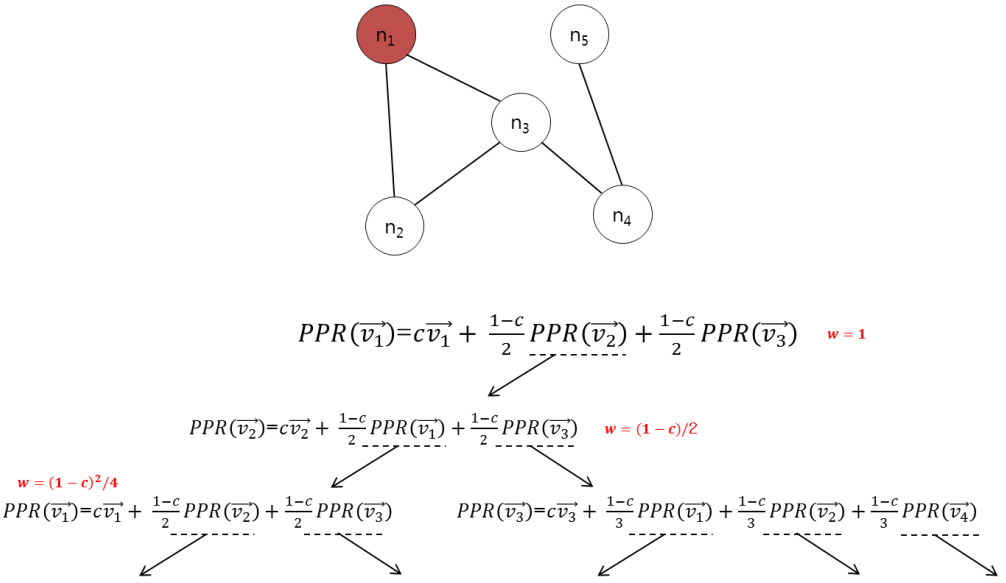


FIGURE 2. Example of bookmark coloring algorithm

Though it can efficiently process queries (one matrix-vector multiplication per each query), it requires massive precomputation time since it performs LU decomposition that requires over $O(N^2)$ time. Considering that the size of the datasets used in their experiments are relatively small (under 300K nodes), it is hard to say whether it can work with very large graphs with over 10M nodes.

BEAR (Block Elimination Approach for Random Walk with Restart on Large Graphs) is one of important recent advancement [27]. It basically follows the studies of the optimized direct equation solving category according to our taxonomy. BEAR reduced the dimension of the matrix that is to be inverted by the block elimination using the Schur complement method [28]. When using the Schur complement method, we need to invert only some submatrices and the Schur complement matrix, which is smaller than the whole matrix.

For an effective application of the Schur complement method, the adjacency matrix should have a large and easy-to-invert submatrix such as a block diagonal matrix. The BEAR utilizes reordering and clustering to easily invert a system matrix, similar to in Fujiwara *et al.* [9]. BEAR decomposed the graph into hubs and spokes. Within each connected component containing spokes, BEAR reorders the nodes in ascending order of the degrees within the component. As a result, BEAR can get an adjacency matrix whose upper-left area is a large and sparse block diagonal matrix that is easily inverted, while the lower-right area is a small but dense matrix.

In the rest of the preprocessing step, BEAR precomputes several matrices including the Schur complement. To solve the equation using the block elimination method, it requires inverting the Schur complement matrix and block diagonal submatrix. BEAR inverts these matrices using LU decomposition. In the query phase, the BEAR quickly computes PPR scores for a given query node using the matrices that are computed in the preprocessing step. The BEAR takes less time and memory space than other preprocessing methods. Their experiments show that their algorithm runs 300 times faster than the simple iterative method.

C. Bookmark Coloring Algorithm

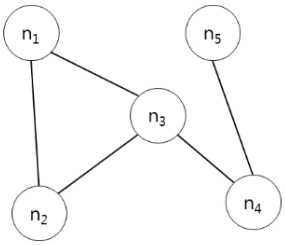
Bookmark Coloring algorithm (BCA) [6] is also an iterative algorithm similar to the power iteration. However, BCA asynchronously updates PPR vectors while the power iteration does it in a synchronous manner. BCA utilize equation (8) in section 2.D. With the interpretation, PPR can be computed in a cumulative way. The basic BCA for computing $PPR(\overrightarrow{v_i})$ can be interpreted using a recursive function $BC(j, w, c)$.

1. Set $\vec{p} \leftarrow c\overrightarrow{v_i}$ when $j = i$, and $\vec{p} \leftarrow \vec{0}$ otherwise.
2. If stopping criterion is met, return $\vec{p}$.
3. For all $n_k \in Out(n_j)$, do $\vec{p} \leftarrow \vec{p} + BC(k, (1-c)w/\deg(n_k), c)$.
4. Return $\vec{p}$.

Fig. 3. illustrates how the basic bookmark coloring algorithm works. It can be represented as a recursive construction of PPR vector by repetitively applying the decomposition theorem.

However, the above algorithm requires a large storage space to store the intermediate PPR vectors. The basic version of the algorithm can be implemented with the following alternative algorithm.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2952653, IEEE Access



Repeat this update until vectors converge

$$PPR^{(k+1)}(\overrightarrow{v_1}) \leftarrow c\overrightarrow{v_1} + \frac{1-c}{2}PPR^{(k)}(\overrightarrow{v_2}) + \frac{1-c}{2}PPR^{(k)}(\overrightarrow{v_3})$$

$$PPR^{(k+1)}(\overrightarrow{v_2}) \leftarrow c\overrightarrow{v_2} + \frac{1-c}{2}PPR^{(k)}(\overrightarrow{v_1}) + \frac{1-c}{2}PPR^{(k)}(\overrightarrow{v_3})$$

$$PPR^{(k+1)}(\overrightarrow{v_3}) \leftarrow c\overrightarrow{v_3} + \frac{1-c}{3}PPR^{(k)}(\overrightarrow{v_2}) + \frac{1-c}{3}PPR^{(k)}(\overrightarrow{v_3}) + \frac{1-c}{3}PPR^{(k)}(\overrightarrow{v_4})$$

$$PPR^{(k+1)}(\overrightarrow{v_4}) \leftarrow c\overrightarrow{v_4} + \frac{1-c}{2}PPR^{(k)}(\overrightarrow{v_3}) + \frac{1-c}{2}PPR^{(k)}(\overrightarrow{v_5})$$

$$PPR^{(k+1)}(\overrightarrow{v_5}) \leftarrow c\overrightarrow{v_5} + (1-c)PPR^{(k)}(\overrightarrow{v_4})$$

FIGURE 3. **Dynamic programming algorithm for the example graph**

1. Initialize $\vec{p} \leftarrow \vec{0}$ and $Q = \{(b, 1)\}$
2. While Q is not empty.
   A. Pop a queue Q element (i, w)
   B. $p_i = p_i + \alpha \cdot w$
   C. If $w < \epsilon$ then continue
   D. For all links $i \rightarrow j \in L$
      i. If $pair(j, s) \in Q$ then $s = s + (1 - \alpha) \cdot w/\deg(i)$.
      ii. Else add a new pair $(j, (1 - \alpha)) \cdot w/\deg(i))$ to Q
3. Return $\vec{p}$.

Though the above algorithm is defined as a recursive function, BCA can also be implemented using a queue data structure [6]. This scheme provides a more efficient implementation utilizing sparsity than matrix-based algorithms such as the power iteration. This version of BCA is also called ForwardPush algorithm [84]. It is possible to compute $PPR(\overrightarrow{v_i})[j]$ for the given target node j and every source node i by reversing the weight propagation direction from the above algorithm. The reversed one is called ReversePush algorithm [84]. Several recent studies combines the Push scheme and Monte-Carlo sampling to improve query processing time. These algorithms are discussed in section 4.E and 4.F.

BCA gradually completes the summation that is defined in equation (8) using repeated asynchronous updates. It can also be viewed as an asynchronous version of the power iteration, and its convergence rate is also identical to the power iteration. That means that BCA requires large numbers of iterations since the basic algorithm cannot benefit from advanced algorithms such as the GMRES in terms of the number of iterations.

If some $PPR(\overrightarrow{v_h})$s are known, BCA can be computed in a more efficient way utilizing the decomposition theorem. We stop the recursive calls and directly construct the return value using known vectors. HubRank [5] is a revised version of BCA with a smart hub selection algorithm and approximation. It provides better performance in terms of the query processing time; however, it still requires a long precomputation time. In an empirical evaluation, it takes over 20 hours to compute PPR vectors for the selected hubs to achieve reasonable query response time.

D. Dynamic Programming

The dynamic programming method for PPR utilizes the decomposition theorem [23]. The most basic algorithm repeatedly updates the process using the decomposition theorem until PPR vectors converge.

1. Set $PPR^{(0)}(\overrightarrow{v_i}) \leftarrow c\overrightarrow{v_i}$ for all n_i.
2. Update $PPR^{(k+1)}(\overrightarrow{v_i}) \leftarrow c\overrightarrow{v_1} + \frac{1-c}{|Out(n_i)|}\sum_{n_j \in O(n_i)} PPR^{(k)}(\overrightarrow{v_j})$ for all n_i.
3. Repeat step 2 until it converges.

Fig. 4. illustrates how the dynamic programming algorithm updates PPR vectors. Like the bookmark coloring algorithm, dynamic programming also updates vectors by repetitively applying the decomposition theorem. However, unlike the bookmark coloring algorithm, dynamic programming updates every $PPR(\overrightarrow{v_i})$ simultaneously.

This algorithm can be applied to the Fully Personalized PageRank problem since it simultaneously produces the $PPR(\overrightarrow{v_i})$s from every node; however, it requires a large space to store the intermediate results. To reduce the space requirement, two methods are applied: rounding and sketching [12].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2952653, IEEE Access

Table 2. Summary of PPR Computation Approaches

	Precomputation Time	Query Answering Time	Auxiliary Data Size	Precision
Iterative Equation Solving	None	High	None	Guaranteed error bound
Direct Equation Solving	Very High	Low	Large	Exact
Bookmark Coloring	None	High	None	Guaranteed error bound
Dynamic Programming	Very High	Low	Large	Guaranteed error bound
Monte-Carlo Sampling	High	Low	Large	Probabilistic error bound

The rounding technique optimizes its space requirement by rounding all values down to a multiple of the prescribed error value ϵ. With the rounding technique, the basic version of dynamic programming changes to the following one.

1. Set $\widehat{PPR}^{(0)}(\overrightarrow{v_i}) \leftarrow c\overrightarrow{v_i}$ for all n_i.
2. Let $\epsilon_k \leftarrow \epsilon \cdot (1-c)$
3. Update $\widehat{PPR}^{(k+1)}(\overrightarrow{v_i}) \leftarrow \psi_k(c\overrightarrow{v_1} + \frac{1-c}{|Out(n_i)|}\sum_{n_j \in O(n_i)} \widehat{PPR}^{(k)}(\overrightarrow{v_j}))$ for all n_i.
4. Repeat steps 2 and 3 until $k = 2\log_{1-c}\epsilon$.

The function ψ_k represents the rounding down to ϵ_k. This algorithm guarantees $PPR(\overrightarrow{v_i})[j] - 2\epsilon/c < \widehat{PPR}(\overrightarrow{v_i})[j] < PPR(\overrightarrow{v_i})[j]$ for all *i*s and *j*s. In the second one, sketching is a hash based approach that utilizes the randomize data structure with low dimensions. It dramatically reduces the space requirement; however, it takes far longer and does not guarantee the error bound. In the empirical evaluation, the dynamic programming algorithm with rounding solves the fully Personalized PageRank problem on a graph with 80M nodes in 2.25 days, and it takes 6 days with rounding and sketching. In summary, these optimization techniques achieve impressive spatial performance gain; however, it is hard to say that they are efficient enough to process large graphs in terms of the time.

E. Monte-Carlo Sampling-based Methods

Monte-Carlo sampling is one traditional method to solve problems with high complexity. It avoids the complexity by deriving the solutions using samples that are produced with a large number of trials. This method can be easily applied to the Personalized PageRank problem. In this case, the sample database is a collection of random walk traces, that is, a set of node sequences (which are called "fingerprints").

There are two Monte-Carlo methods for the Personalized PageRank problem: the MC end-point and the MC complete path [22]. The MC end-point method uses the summation form (equation (8)). If we know $P^k\vec{s}$ with enough depth *k*,

We can compute $PPR(\vec{s})$ using simple algebraic operations, and $P^k\vec{s}$ can be approximated by counting the end points of the sample traces with length *k* from the starting node that is specified within $\overrightarrow{s}$. While the MC end-point uses only the end point of each trace, the MC complete path method uses every node within traces.

Monte-Carlo scheme is mainly utilized in the fully Personalized PageRank problem since it does not requires a square space to store all computed PPR vectors. One of most important algorithms in this category is doubling [4]. It populates long traces by concatenating short traces, and the procedure is defined using MapReduce framework. In the empirical evaluation, the doubling algorithm performs approximately 8 times faster than rounding (section 4.E). In spite of its impressive enhancement, it does not strictly guarantee the error bound because of the substantial limitation of sampling method, and it is hard to say that it obviously outperforms other advanced approaches considering that rounding is a relatively underperforming algorithm that has an identical convergence rate to the power iteration.

One of recent studies, FAST-PPR, also adopts the Monte-Carlo scheme. It solves point-to-point Personalized PageRank problem that computes $PPR(\overrightarrow{v_i})[j]$ when the graph and two nodes (source and sink) are given [11]. It utilizes two sided random walk samples (from the source and sink) at the same time. It can be viewed as the combination of Monte-Carlo sampling and ReversePush that is discussed in section 4.C. As a result, it achieves impressive performance. However, it is unclear how to utilize the point-to-point PPR in the applications since the entries of PPR vectors are relative values. Another recently proposed method, PowerWalk [88] is also a sampling-based algorithm. To produce accurate answers, it process queries in iterative manners based on the sampled information.

F. Other Methods

There are several studies that cannot be categorized into the above categories. The study on finding the top-k nearest nodes based on PPR is one of the most important examples [8]. In this study, they propose a unified algorithm to solve the top-k problem for several random walk-based node similarity (i.e., proximity) measures including the SimRank [24], Discounted Hitting Time, and PPR. First, they define the unified form generalizing those proximity measures between two nodes n_1 and n_2 as follows:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2952653, IEEE Access

Table 3. Summary of PPR Computation Algorithms

Category	Publication info	Feature	Auxiliary Data	Precision
Direct Equation Solving	ICDM 2006 [1]	Low rank approximation	Approximated inverse matrix	Inexact
	VLDB 2011 [9]	Matrix inversion after reordering	Decomposed inverse matrix	Exact
	SIGIR 2013 [83]	Updating precomputed PPR vectors in direct equation solving scheme	Not required	Exact
	SIGMOD 2015 [27]	Matrix inversion after block elimination	Decomposed inverse matrix	Exact
	SIGMOD 2017 [58]	Matrix inversion after block elimination, and fine tuning the result with iterative methods	Decomposed inverse matrix	Inexact result with guaranteed error bound
Iterative Equation Solving	WWW 2003 [30]	Accelerated power iteration via extrapolation.	Not required	Inexact result with guaranteed error bound
	KDD 2010 [25]	Turning the nodes with high out-degree into sinks to reduce computation cost.	Not required	Inexact
	VLDB 2014 [10]	Iterative method with preconditioning by core-tree decomposition	Result of core-tree decomposition	Inexact result with guaranteed error bound
	KDD 2016 [84]	A variant of power iteration for dynamic graphs	Not required	Inexact
	WWW 2018 [85]	A variant of power iteration for dynamic graphs	Not required	Inexact result with guaranteed error bound
Bookmark Coloring Algorithm	Internet Math. 2006 [6]	Basic bookmark coloring algorithm (BCA)	Not required	Inexact result with guaranteed error bound
	VLDB Journal 2011, WWW 2007 [5]	Revised BCA using precomputed PPR vectors of selected nodes	Precomputed PPR vectors of selected query nodes	Inexact
	KDD 2015 [86]	Updating precomputed PPR vectors by BCA-like procedure	Not required	Inexact result with guaranteed error bound
Dynamic Programming	WWW 2003 [23]	Basic dynamic programming for PPR	Precomputed PPR vectors of selected query nodes	Inexact result with guaranteed error bound
	WWW 2006 [12]	Dynamic programming with rounding & sketch	All PPR vectors need to be stored while processing	Inexact
Monte-Carlo Sampling	Internet Math. 2005 [22]	Basic Monte-Carlo sampling for PPR	Sampled random walk traces	Inexact
	SIGMOD 2011 [4]	Monte-Carlo sampling with doubling	Sampled random walk traces	Inexact
	KDD 2014 [11]	Combining Monte-Carlo sampling and ReversePush	Not required	Inexact
	CIKM 2016 [88]	Sampling-based method with iterative query processing	Sampled random walk traces	Inexact
Other	VLDB 2011 [8]	Top-k query processing by error bound estimation	Not required	Exact top-k list
	ICML 2014 [26]	Anti-differentiating approximation	Not required	Exact
	VLDB Journal 2015 [29]	Scheduling with hub info	Tours and reachability information	Inexact
	WSDM 2016 [80]	Top-k query processing by error bound estimation, and combination of Monte-Carlo sampling and ReversePush	Sampled random walk traces	Inexact top-k list
	VLDB 2016 [81]	Top-k query processing by error bound estimation, and combination of Monte-Carlo sampling and ReversePush	Sampled random walk traces	Inexact top-k list
	KDD 2017 [82]	Top-k query processing by error bound estimation, and combination of Monte-Carlo sampling and ForwardPush	Sampled random walk traces	Inexact top-k list

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2952653, IEEE Access

$$S(n_i, n_j) = \lim_{d \to \infty} S_d(n_1, n_2)$$
$$\text{where } S_d(n_i, n_j) = a \sum_{k=1}^{d} \lambda^k P_k(n_i, n_j) + b \quad (11)$$

Here, a and b are real-valued constants ($a > 0$), and $\lambda \in (0,1)$. For PPR, $a = c, \lambda = 1 - c, P_k(n_i, n_j) = P^k \vec{v_i}[j]$, and $b = 0$. From the above formulation, we can compute the upper bound of $S(n_i, n_j)$ as follows:

$$S(n_i, n_j) \in [S_d(n_i, n_j), S_d(n_i, n_j) + X_d^+]$$
$$\text{where } X_d^+ = \frac{a \cdot \lambda^{d+1}}{1 - \lambda} \quad (12)$$

Using the above upper bound information, we can prune out unpromising nodes in the early stages of the algorithm. This algorithm is impressive in terms of time and space. It runs fast and requires no auxiliary data structure. The biPPR [80], HubPPR [81], and FORA [82] solve the same problem. Like the previous algorithm, they also estimate bound information to reduce computation cost. These algorithms are based on the combination of the push scheme and Monte-Carlo sampling like FAST-PPR [11]. The biPPR and HubPPR adopt ReversePush while FORA adopts ForwardPush.

There are some optimization techniques such as turning the nodes with high out-degree into sinks to reduce computation time with low approximation error [25] and introducing novel approximation schemes such as anti-differentiating approximation algorithms [26].

G. Recent Advances

The efficient computation of PPR is still one of major on-going issues of massive graph processing, and many remarkable studies have been published recently.

BePI [58] is a study that combines the preprocessing method and the iterative method based on the BEAR. BePI addresses the challenges that are faced by previous approaches by combining the best of both the preprocessing and iterative methods. BePI uses a block elimination approach, which is a preprocessing method, to achieve a fast query time. BePI incorporates an iterative method within the block elimination to decrease the memory requirements by avoiding expensive matrix inversions. BePI takes the advantages of both the preprocessing methods and iterative methods. Consequently, BePI achieves a better scalability and a faster query time than other methods.

FastPPV (different from the Fast-PPR [11] that was introduced in section 4.E) is also a remarkable innovation for this problem [29][54]. The study presented a scheduled approximation strategy to approximate PPR vectors. Specifically, they developed a hub based scheduling scheme and a structured aggregation model. They also explored the issue of hub selection. Their experiment results show that their method outperforms HubRank and Monte-Carlo sampling methods.

Multiple methods focusing on dealing with dynamic graphs have been suggested [83][84][85][86]. They update the precomputed PPR vectors when the edges are added or deleted, and they also can be categorized into major approaches like other works. IRWR [83] updates PPR vectors based on direct solving formula, and LayFwdUpdate [84], and OSP [85] are dynamic variants of iterative methods. TrackingPPR [86] works on the bookmark coloring scheme.

V. Discussions

In the previous section, we review the major studies on the computation of PPR. Each approach has distinctive characteristics, and one should choose the most appropriate algorithm by considering the most suitable characteristics to properly utilize PPR. Table 2 summarizes the characteristics of the five approaches that presented in the previous section.

In terms of the precision, direct equation solving is the best algorithm. If errors are totally unacceptable, direct equation solving is the only valid solution. Though their precomputation costs are high, the advanced techniques based on approaches such as the BEAR reduce the large amount of precomputation time. Iterative equation solving, the Bookmark Coloring algorithm, and Dynamic Programming cannot produce exact answers; however, their errors can be controlled since they guarantee a predefined error bound. With a very small error bound, the outputs can be viewed as near exact answers. One should consider using approaches other than exact algorithms since a large portion of the applications beside scientific data analysis do not require exact computations of PPR.

When an application requires short precomputation time, direct equation solving-based techniques and Monte-Carlo sampling can be considered as improper solutions. In contrast, iterative methods and Bookmark Coloring do not require precomputation phases. Though iterative methods require longer query time computations, this can be overcome by adopting an advanced iterative scheme such as preconditioning and overrelaxation. Some advanced techniques based on approaches such as HubRank reduce the query time computations by introducing precomputation phases. One should consider these techniques when the query time computation is problematic.

The auxiliary data size also should be considered. Direct equation solving requires space to store the matrix inversion, and Monte-Carlo sampling must store large numbers of random walk traces to achieve high precision while iterative methods and Bookmark Coloring require no auxiliary data space. If no additional storage to support PPR computation is available, the most basic iterative methods and Bookmark Coloring algorithms should be introduced. However, there exist several advanced techniques that utilize additional space to reduce the computation time. If the situation is not extremely limited in terms of space, one should consider those advanced algorithms.

VI. Conclusion

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2952653, IEEE Access

We have provided a comprehensive summary of the current studies on computing PPR in massive graphs by categorizing them into five major approaches: iterative equation solving, direct equation solving, bookmark coloring algorithm, dynamic programming, and Monte-Carlo sampling.

Future research direction can be derived from the recent successful studies. One of recently proposed algorithm, BePI combines the direct equation solving and the iterative equation solving. FAST-PPR, biPPR, hubPPR and FORA improves the performance by combining Monte-Carlo sampling and BCA-like scheme. Since each approach has been highly matured through years, it is a proper approach to combine the multiple schemes and utilize their merits.

Fusing with less highlighted approach such as iterative method can be a promising option. BePI can be the example, and the continuing advancements in iterative equation solvers [87] make the iterative approach an important candidate to consider.

Sticking to one dominant approach is not the ideal solution. Comprehending the characteristics of diverse approaches and effectively utilizing them can greatly contribute to making significant breakthroughs. We hope that this survey to be a meaningful source of insights for the future research on PPR computation.

Acknowledgement

This research is supported by Ministry of Culture, Sports and Tourism(MCST) and Korea Creative Content Agency(KOCCA) in the Culture Technology(CT) Research & Development Program 2019 (R2019050030)

References

[1] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in Sixth International Conference on Data Mining (ICDM'06), pp. 613–622, IEEE, 2006.

[2] D. Fogaras, B. R´acz, K. Csalog´any, and T. Sarl´os, "Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments," Internet Mathematics, vol. 2, no. 3, pp. 333–358, 2005.

[3] D. Fogaras and B. R´acz, "Towards scaling fully personalized pagerank," in International Workshop on Algorithms and Models for the Web-Graph, pp. 105–117, Springer, 2004.

[4] B. Bahmani, K. Chakrabarti, and D. Xin, "Fast personalized pagerank on mapreduce," in Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, pp. 973–984, ACM, 2011.

[5] S. Chakrabarti, A. Pathak, and M. Gupta, "Index design and query processing for graph conductance search," The VLDB Journal, vol. 20, no. 3, pp. 445–470, 2011.

[6] P. Berkhin, "Bookmark-col

[7] T. H. Haveliwala, "Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search," IEEE transactions on knowledge and data engineering, vol. 15, no. 4, pp. 784–796, 2003.

[8] L. Sun, C. Cheng, X. Li, D. Cheung, and J. Han, "On link-based similarity join," Proceedings of the VLDB Endowment, 2011.

[9] Y. Fujiwara, M. Nakatsuji, M. Onizuka, and M. Kitsuregawa, "Fast and exact top-k search for random walk with restart," Proceedings of the VLDB Endowment, vol. 5, no. 5, pp. 442–453, 2012.

[10] T. Maehara, T. Akiba, Y. Iwata, and K.-i. Kawarabayashi, "Computing personalized pagerank quickly by exploiting graph structures," Proceedings of the VLDB Endowment, vol. 7, no. 12, pp. 1023–1034, 2014.

[11] P. A. Lofgren, S. Banerjee, A. Goel, and C. Seshadhri, "Fast-ppr: Scaling personalized pagerank estimation for large graphs," in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1436–1445, ACM, 2014.

[12] T. Sarl´os, A. A. Bencz´ur, K. Csalog´any, D. Fogaras, and B. R´acz, "To randomize or not to randomize: space optimal summaries for hyperlink analysis," in Proceedings of the 15th international conference on World Wide Web, pp. 297–306, ACM, 2006.

[13] Y. Z. Guo, K. Ramamohanarao, and L. A. Park, "Personalized pagerank for web page prediction based on access time-length and frequency," in IEEE/WIC/ACM International Conference on Web Intelligence (WI'07), pp. 687–690, IEEE, 2007.

[14] F. Pedroche, F. Moreno, A. Gonz´alez, and A. Valencia, "Leadership groups on social network sites based on personalized pagerank," Mathematical and Computer Modelling, vol. 57, no. 7-8, pp. 1891–1896, 2013.

[15] E. Agirre and A. Soroa, "Personalizing pagerank for word sense disambiguation," in Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, pp. 33–41, Association for Computational Linguistics, 2009.

[16] Y. Liu, X. Wang, J. Zhang, and H. Xu, "Personalized pagerank based multidocument summarization," in IEEE International Workshop on Semantic Computing and Systems, pp. 169–173, IEEE, 2008.

[17] H.-N. Kim and A. El Saddik, "Personalized pagerank vectors for tag recommendations: inside folkrank," in Proceedings of the fifth ACM conference on Recommender systems, pp. 45–52, ACM, 2011.

[18] S. Lee, S. Park, M. Kahng, and S.-g. Lee, "Pathrank: a novel node ranking measure on a heterogeneous graph for recommender systems," in Proceedings of the 21st ACM international conference on Information and knowledge management, pp. 1637–1641, ACM, 2012.

[19] T. H. Kim, K. M. Lee, and S. U. Lee, "Generative image segmentation using random walks with restart," in European conference on computer vision, pp. 264–275, Springer, 2008. 2

[20] B. Ham, D. Min, and K. Sohn, "A generalized random walk with restart and its application in depth up-sampling and interactive segmentation," IEEE Transactions on Image Processing, vol. 22, no. 7, pp. 2574–2588, 2013.

[21] G. Iv´an and V. Grolmusz, "When the web meets the cell: using personalized pagerank for analyzing protein interaction networks," Bioinformatics, vol. 27, no. 3, pp. 405–407, 2010.

[22] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova, "Monte carlo methods in pagerank computation: When one iteration is sufficient," SIAM Journal on Numerical Analysis, vol. 45, no. 2, pp. 890–904, 2007.

[23] G. Jeh and J. Widom, "Scaling personalized web search," in Proceedings of the 12th international conference on World Wide Web, pp. 271–279, Acm, 2003.

[24] G. Jeh and J. Widom, "Simrank: a measure of structural-context similarity," in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 538–543, ACM, 2002.

[25] P. Sarkar and A. W. Moore, "Fast nearest-neighbor search in disk-resident graphs," in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 513–522, ACM, 2010.

[26] D. Gleich and M. Mahoney, "Anti-differentiating approximation algorithms: A case study with min-cuts, spectral, and flow," in International Conference on Machine Learning, pp. 1018–1025, 2014.

[27] K. Shin, J. Jung, S. Lee, and U. Kang, "Bear: Block elimination approach for random walk with restart on large graphs," in Proceedings of the 2015 ACM SIGMOD international conference on Management of Data, pp. 1571–1585, ACM, 2015.

[28] S. Boyd and L. Vandenberghe, Convex optimization. Cambridge university press, 2004.

[29] F. Zhu, Y. Fang, K. C.-C. Chang, and J. Ying, "Scheduled approximation for personalized pagerank with utility-based hub selection," The VLDB Journal—The International Journal on Very Large Data Bases, vol. 24, no. 5, pp. 655–679, 2015.

[30] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, "Extrapolation methods for accelerating pagerank computations," in Proceedings of the 12th international conference on World Wide Web, pp. 261–270, ACM, 2003.

[31] A. N. Langville and C. D. Meyer, Google's PageRank and beyond: The science of search engine rankings. Princeton University Press, 2011.
[32] L. Page, S. Brin, R. Motwani, and T. Winograd, "PageRank citation ranking: Bringing order to the web", tech. rep., Stanford InfoLab, 1999.
[33] Z. Junchao, C. Junjie, J. Song, and R.-X. Zhao, "Monte carlo based personalized pagerank on dynamic networks," International Journal of Distributed Sensor Networks, vol. 9, no. 9, p. 829804, 2013.
[34] C. Borgs, M. Brautbar, J. Chayes, and S.-H. Teng, "Multiscale matrix sampling and sublinear-time pagerank computation," Internet Mathematics, vol. 10, no. 1-2, pp. 20–48, 2014.
[35] I. Konstas, V. Stathopoulos, and J. M. Jose, "On social networks and collaborative recommendation," in Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, pp. 195–202, ACM, 2009.
[36] S. Lee, S.-i. Song, M. Kahng, D. Lee, and S.-g. Lee, "Random walk based entity ranking on graph for multidimensional recommendation," in Proceedings of the fifth ACM conference on Recommender systems, pp. 93–100, ACM, 2011.
[37] B. Bahmani, A. Chowdhury, and A. Goel, "Fast incremental and personalized pagerank," Proceedings of the VLDB Endowment, vol. 4, no. 3, pp. 173–184, 2010.
[38] A. Balmin, V. Hristidis, and Y. Papakonstantinou, "Objectrank: Authority-based keyword search in databases," in Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, pp. 564– 575, VLDB Endowment, 2004.
[39] Y. Fujiwara, M. Nakatsuji, T. Yamamuro, H. Shiokawa, and M. Onizuka, "Efficient personalized pagerank with accuracy assurance," in Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 15–23, ACM, 2012.
[40] S. Chakrabarti, "Dynamic personalized pagerank in entity-relation graphs," in Proceedings of the 16th international conference on World Wide Web, pp. 571–580, ACM, 2007.
[41] S. Al-Saffar and G. Heileman, "Experimental bounds on the usefulness of personalized and topic-sensitive pagerank," in Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, pp. 671– 675, IEEE Computer Society, 2007.
[42] W. Y. Wang, K. Mazaitis, and W. W. Cohen, "Programming with personalized pagerank: a locally groundable first-order probabilistic logic," in Proceedings of the 22nd ACM international conference on Information & Knowledge Management, pp. 2129–2138, ACM, 2013.
[43] D. Gleich and M. Polito, "Approximating personalized pagerank with minimal use of web graph data," Internet Mathematics, vol. 3, no. 3, pp. 257–294, 2006.
[44] J. Xia, D. Caragea, and W. H. Hsu, "Bi-relational network analysis using a fast random walk with restart," in 2009 Ninth IEEE International Conference on Data Mining, pp. 1052–1057, IEEE, 2009. 4
[45] H. Tong and C. Faloutsos, "Center-piece subgraphs: problem definition and fast solutions," in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 404–413, ACM, 2006.
[46] Z. Yin, M. Gupta, T. Weninger, and J. Han, "Linkrec: a unified framework for link recommendation with user attributes and graph structure," in Proceedings of the 19th international conference on World wide web, pp. 1211–1212, ACM, 2010.
[47] 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, no. 1, p. 283, 2009.
[48] X. Wang, N. Gulbahce, and H. Yu, "Network-based methods for human disease gene prediction," Briefings in functional genomics, vol. 10, no. 5, pp. 280–293, 2011.
[49] W. Yu and X. Lin, "Irwr: incremental random walk with restart," in Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, pp. 1017–1020, ACM, 2013.
[50] R. Andersen, F. Chung, and K. Lang, "Local partitioning for directed graphs using pagerank," Internet Mathematics, vol. 5, no. 1-2, pp. 3–22, 2008.
[51] Z. Dou, R. Song, and J.-R. Wen, "A large-scale evaluation and analysis of personalized search strategies," in Proceedings of the 16th international conference on World Wide Web, pp. 581–590, ACM, 2007.
[52] S. A. Tabrizi, A. Shakery, M. Asadpour, M. Abbasi, and M. A. Tavallaie, "Personalized pagerank clustering: A graph clustering algorithm based on random walks," Physica A: Statistical Mechanics and its Applications, vol. 392, no. 22, pp. 5772–5785, 2013.
[53] A. Z. Broder, R. Lempel, F. Maghoul, and J. Pedersen, "Efficient pagerank approximation via graph aggregation," Information Retrieval, vol. 9, no. 2, pp. 123–138, 2006.
[54] F. Zhu, Y. Fang, K. C.-C. Chang, and J. Ying, "Incremental and accuracyaware personalized pagerank through scheduled approximation," Proceedings of the VLDB Endowment, vol. 6, no. 6, pp. 481–492, 2013.
[55] E. Garcia, F. Pedroche, and M. Romance, "On the localization of the personalized pagerank of complex networks," Linear Algebra and its Applications, vol. 439, no. 3, pp. 640–652, 2013.
[56] G. M. Del Corso, A. Gulli, and F. Romani, "Fast pagerank computation via a sparse linear system," Internet Mathematics, vol. 2, no. 3, pp. 251–273, 2005.
[57] A. N. Langville and C. D. Meyer, "Deeper inside pagerank," Internet Mathematics, vol. 1, no. 3, pp. 335–380, 2004. 5
[58] J. Jung, N. Park, S. Lee, and U. Kang, "Bepi: Fast and memory-efficient method for billion-scale random walk with restart," in Proceedings of the 2017 ACM International Conference on Management of Data, pp. 789–804, ACM, 2017.
[59] L. Dong, Y. Li, H. Yin, H. Le, and M. Rui, "The algorithm of link prediction on social network," Mathematical Problems in Engineering, vol. 2013, 2013.
[60] L. Backstrom and J. Leskovec, "Supervised random walks: predicting and recommending links in social networks," in Proceedings of the fourth ACM international conference on Web search and data mining, pp. 635–644, ACM, 2011.
[61] W. Liu and L. L¨u, "Link prediction based on local random walk," EPL (Europhysics Letters), vol. 89, no. 5, p. 58007, 2010.
[62] J. Jung, W. Jin, L. Sael, and U. Kang, "Personalized ranking in signed networks using signed random walk with restart," in 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 973–978, IEEE, 2016.
[63] R. Devooght, A. Mantrach, I. Kivim¨aki, H. Bersini, A. Jaimes, and M. Saerens, "Random walks based modularity: application to semisupervised learning," in Proceedings of the 23rd international conference on World wide web, pp. 213–224, ACM, 2014.
[64] M. Pershina, Y. He, and R. Grishman, "Personalized page rank for named entity disambiguation," in Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 238–243, 2015.
[65] C. Wang, F. Jing, L. Zhang, and H.-J. Zhang, "Image annotation refinement using random walk with restarts," in Proceedings of the 14th ACM international conference on Multimedia, pp. 647–650, ACM, 2006.
[66] J.-S. Kim, J.-Y. Sim, and C.-S. Kim, "Multiscale saliency detection using random walk with restart," IEEE transactions on circuits and systems for video technology, vol. 24, no. 2, pp. 198–210, 2013.
[67] H. Kim, Y. Kim, J.-Y. Sim, and C.-S. Kim, "Spatiotemporal saliency detection for video sequences based on random walk with restart," IEEE Transactions on Image Processing, vol. 24, no. 8, pp. 2552–2564, 2015.
[68] S. Lee, J. H. Lee, J. Lim, and I. H. Suh, "Robust stereo matching using adaptive random walk with restart algorithm," Image and Vision Computing, vol. 37, pp. 1–11, 2015.
[69] T. H. Kim, K. M. Lee, and S. U. Lee, "Edge-preserving colorization using data-driven random walks with restart," in 2009 16th IEEE international conference on image processing (ICIP), pp. 1661–1664, IEEE, 2009.
[70] C. Oh, B. Ham, and K. Sohn, "Probabilistic correspondence matching using random walk with restart.," in BMVC, pp. 1–10, 2012. 6
[71] J. Sun, H. Shi, Z. Wang, C. Zhang, L. Liu, L. Wang, W. He, D. Hao, S. Liu, and M. Zhou, "Inferring novel lncrna–disease associations based on a random walk model of a lncrna functional similarity network," Molecular BioSystems, vol. 10, no. 8, pp. 2074–2081, 2014.

[72] X. Chen, Z.-H. You, G.-Y. Yan, and D.-W. Gong, "Irwrlda: improved random walk with restart for lncrna-disease association prediction," Oncotarget, vol. 7, no. 36, p. 57919, 2016.
[73] X. Chen, M.-X. Liu, and G.-Y. Yan, "Drug–target interaction prediction by random walk on the heterogeneous network," Molecular BioSystems, vol. 8, no. 7, pp. 1970–1978, 2012.
[74] J. Li, L. Chen, S. Wang, Y. Zhang, X. Kong, T. Huang, and Y.-D. Cai, "A computational method using the random walk with restart algorithm for identifying novel epigenetic factors," Molecular genetics and genomics, vol. 293, no. 1, pp. 293–301, 2018.
[75] C. Blatti and S. Sinha, "Characterizing gene sets using discriminative random walks with restart on heterogeneous biological networks," Bioinformatics, vol. 32, no. 14, pp. 2167–2175, 2016.
[76] K. C. Chipman and A. K. Singh, "Predicting genetic interactions with random walks on biological networks," BMC bioinformatics, vol. 10, no. 1, p. 17, 2009.
[77] M. Nykl, M. Campr, and K. Je˘zek, "Author ranking based on personalized pagerank," Journal of Informetrics, vol. 9, no. 4, pp. 777–799, 2015.
[78] H. Avron and L. Horesh, "Community detection using time-dependent personalized pagerank," in International Conference on Machine Learning, pp. 1795–1803, 2015.
[79] Piegorsch and G. E. Casella. Inverting a sum of matrices. In SIAM Review, 1990.
[80] Lofgren, Peter, Siddhartha Banerjee, and Ashish Goel. "Personalized pagerank estimation and search: A bidirectional approach." Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. pp. 163-172, 2016..
[81] Wang, S., Tang, Y., Xiao, X., Yang, Y., and Li, Z. "HubPPR: effective indexing for approximate personalized pagerank." Proceedings of the VLDB Endowment, vol. 10, no. 3 pp. 205-216, 2016
[82] Wang, S., Yang, R., Xiao, X., Wei, Z., and Yang, Y., "Fora: Simple and effective approximate single-source personalized pagerank." In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 505-514, 2017.
[83] Yu, W., and Lin, X., "IRWR: incremental random walk with restart." In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval, pp. 1017-1020, 2013.
[84] Zhang, H., Lofgren, P., and Goel, A., "Approximate personalized pagerank on dynamic graphs." In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1315-1324, 2016.
[85] Yoon, M., Jin, W., and Kang, U., "Fast and accurate random walk with restart on dynamic graphs with guarantees." In Proceedings of the 2018 World Wide Web Conference, pp. 409-418, 2018.
[86] Ohsaka, N., Maehara, T., and Kawarabayashi, K. I., "Efficient pagerank tracking in evolving networks." In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 875-884, 2015.
[87] Adsuara, J. E., Cordero-Carrión, I., Cerdá-Durán, P., and Aloy, M. A., "Scheduled relaxation Jacobi method: improvements and applications." Journal of Computational Physics, Vol. 321, pp. 369-413, 2016.
[88] Liu, Q., Li, Z., Lui, J., and Cheng, J., "Powerwalk: Scalable personalized pagerank via random walks with vertex-centric decomposition." In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp. 195-204, 2016.