IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 24, 2018, accepted June 21, 2018, date of publication June 29, 2018, date of current version August 20, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2851604

Fast PageRank Computation Based on Network Decomposition and DAG Structure

ZHIBO ZHU[1], QINKE PENG[1], ZHI LI[2], XINYU GUAN[1], AND OWAIS MUHAMMAD[1]
[1]Systems Engineering Institute, Xi'an Jiaotong University, Xi'an 710049, China
[2]Works Applications China Company Ltd., Shanghai 200062, China

Corresponding author: Qinke Peng (qkpeng@mail.xjtu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61173111 and in part by the Ph.D. Programs Foundation of Ministry of Education of China under Grant 20090201110027.

ABSTRACT PageRank has been widely used for the problem of evaluating the importance of data in many applications, such as Web science, information systems, and social network analysis. As vast amounts of data generate, the development of efficient PageRank computation is a vibrant area of contemporary research. In this paper, we propose a fast PageRank computation method based on network decomposition and the structure of directed acyclic graph (DAG). A network decomposition technique is first introduced to decompose the original network into three parts, including a general sub-network and two sub-DAGs. Based on the acyclic characteristic, we demonstrate that these two sub-DAGs can be theoretically lumped into two single nodes by a similarity transformation. Then, the PageRank problem on the original network is transformed into a PageRank problem on a much smaller network, and the full PageRank vector can be easily recovered from the result of new problem. As the time taken by the estimation of PageRank is directly proportional to the network size, the proposed method achieves an improved time complexity and is more efficient as the size of two sub-DAGs increases. Experimental results on real data sets show that our method provides a significant speedup compared with existing alternatives.

INDEX TERMS PageRank, link analysis, DAG structure, network decomposition.

I. INTRODUCTION

In recent years, many information systems are modeled as complex networks, where nodes represent the data objects and edges specify the relationships among them. Problems, such as finding the most influencing node or top-K influential nodes in a network, arise in plenty of data mining applications including information retrieval [1], [2], literature analysis [3]–[5], and recommendation systems [6]. Many different methods have been studied to address these problems by estimating the overall importance of nodes and ranking as per their importance score [7]–[10]. Among those, PageRank is one of the most popular measures, due to its successful application in search engine and clear theoretical foundation.

The development of methods for efficient PageRank computation is a vibrant area of contemporary research. The representative algorithm is the power method, which stands out for the stable and reliable performance [11]. However, as an iterative algorithm, the power method calculates the PageRank values of all nodes in each of its iterations. Hence its application to large networks can take a considerable time to yield results, such as web graph and citation network. Another drawback of the power method is the indeterminate convergence time, closely depending on the structure of network itself and the prescribed parameters [12], [13].

Many studies aim to accelerate the PageRank computation to remedy the shortcomings of the power method [14], [15]. The extrapolation method reduces the computation cost via periodically subtracting off the estimates of nonprincipal eigenvectors in the iteration [16]–[18]. The adaptive method speeds up the algorithm with no recomputation for the converged PageRank values, according to the discovery that the convergence pattern of all PageRank values has a nonuniform distribution [19]. The distributed randomized algorithms are proposed based on the multi-agent consensus [20], [21], and there also exist some advanced numerical computation methods [22], [23].

Besides, some researchers paid attention to some special kind of nodes in a network to reduce the size of system solved by iterative computation. Ipsen and Selee [24] lumped the dangling nodes in a network to improve the performance. Lin *et al.* [25] improved the above method by lumping two classes of nodes, including dangling nodes and weakly dangling nodes, which only reach to dangling nodes. Based on that, Yu *et al.* [26] gave a unified presentation of a few lumping methods for the PageRank computation. In addition, Langville and Meyer [27] proposed a reordered PageRank algorithm, followed by Bu and Huang [28] who introduced an adaptive reordered method for further speedup.

The existing methods are primarily based on the iterative process which demands a set of iterations to converge. Considering the scope of the PageRank problem, reducing even a handful of iterations is praiseworthy. This paper introduces a method for fast PageRank computation based on network decomposition and DAG (directed acyclic graph) structures, which is essentially distinguished from other methods. We firstly decompose the original network into three parts, including two sub-DAGs and a general sub-network. Owe to the acyclic characteristic of DAGs, such as food web [29] and disease network [30], [31], it is feasible to lump two sub-DAGs into two single nodes respectively. Accordingly, the final PageRank is obtained by solving problem on a much smaller network, leading to a great reduction of the iteration cost.

The remainder of this paper is organized as follows. In section II, we provide an overview of the concepts of PageRank. Section III presents the proposed method for fast PageRank computation together with its characteristics. The experiments and case-study are included in section IV. Concluding remarks can be found in section V.

II. PRELIMINARIES

PageRank was originally proposed to rank web pages by assigning an importance score for each page [2]. The basic idea is that the importance of any page in the web graph is in relation to the quantity and the quality of the pages linking to it. Specifically, the importance score of one web page depends on two aspects: the number of its incoming hyperlinks and the importance scores of web pages linking to it.

Assume a web graph is represented as a directed network $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ with $n \in \mathbb{N}$ is the set of nodes representing pages, and $E = \{(i, j)|$ node i points to j; $i, j \in V\}$ is the set of edges as hyperlinks. $A = (a_{ij})$ is the adjacency matrix of G such that $a_{ij} = 1$ if $(i, j) \in E$ and 0 otherwise. Let $B_i = \{j|(j, i) \in E\}$ is the set of parent nodes, and $F_i = \{j|(i, j) \in E\}$ is the set of child nodes of node i respectively. Then the PageRank value p_i of node i is formalized as below,

$$p_i = \sum_{j \in B_i} \frac{p_j}{|F_j|} \tag{1}$$

The matrix form of the above iterative definition is following,

$$p^{\mathrm{T}} = p^{\mathrm{T}} P \tag{2}$$

where $p = [p_1, \ldots, p_n]^{\mathrm{T}}$ is the PageRank vector, and $P = D^{-1}A$ with $D = \mathrm{diag}(\max\{|F_1|, 1\}, \max\{|F_2|, 1\}, \ldots, \max\{|F_n|, 1\})$ is the initial transition matrix.

If there are no out-links from node i, namely $|F_i| = 0$, the transition matrix P does not satisfy the necessary condition being row-stochastic. These nodes are dangling nodes, which lead to a computation problem for PageRank. The most popular remedy is uniformly connecting dangling nodes to all nodes [12], namely

$$P' = P + c \cdot v^T \tag{3}$$

where $v = [1/n, \ldots, 1/n]^{\mathrm{T}}$ is the teleportation vector, and $c = [c_1, \ldots, c_n]^{\mathrm{T}}$ in which c_i is 1 if i is a dangling node and 0 otherwise.

In addition, another possible problem is that some loops may result in the rank sink in a network [11]. Considering two nodes i and j that are pointing to each other only, while node k is pointing to node i. Then during the computation process, the PageRank values would be accumulated in this loop only, and never distributed to other nodes as there are no out-links to other nodes. The remedy for this problem of rank silk is analogous to the modification for dangling nodes, and the final transition matrix P'' is as below,

$$P'' = dP' + (1 - d)e \cdot v^{\mathrm{T}} \tag{4}$$

where $0 < d < 1$ is the damping factor and $e = [1, \ldots, 1]^{\mathrm{T}}$.

This transformation retains the row-stochastic property of the transition matrix. Finally, the PageRank vector is the solution of the following eigenvector problem,

$$p^{\mathrm{T}} = p^{\mathrm{T}} P'' = dp^{\mathrm{T}} P' + (1 - d)v^{\mathrm{T}} \tag{5}$$

The definition of PageRank can be described as a random walk model [32]: a walker starts at a uniformly chosen random node, 1) from the dangling node, it teleports to a random node; 2) from other node, it follows a uniformly selected out-links with probability d, or teleports to a random node with probability $1 - d$. The PageRank value eventually means the probability the walker visiting each node. Besides the uniform vector, the teleportation vector v can be any nonnegative vector with $||v||_1 = 1$, known as the personalization vector in many applications [33]–[35]. The proposed method in this paper is appropriate to any form of the teleportation vector.

III. FAST PAGERANK

There are many real systems modeled by directed networks, such as web graphs, biological networks, and social networks [36]–[39]. Besides the cycles, there also exist some sub-DAGs in these networks, especially the web graphs with the bowtie structure. The intuition behind our method is that, this special structure of sub-DAGs can minimize the number of nodes, whose PageRank values need to be iteratively computed. More specifically, if we can decompose the sub-DAGs from the whole network and transform the original PageRank problem into a smaller one, the cost for iterative computation would be largely reduced such that improving the efficiency of PageRank estimation. Fig.1 shows the framework of our proposed fast PageRank method. In this section, we firstly explain the network decomposition mechanism, then describe

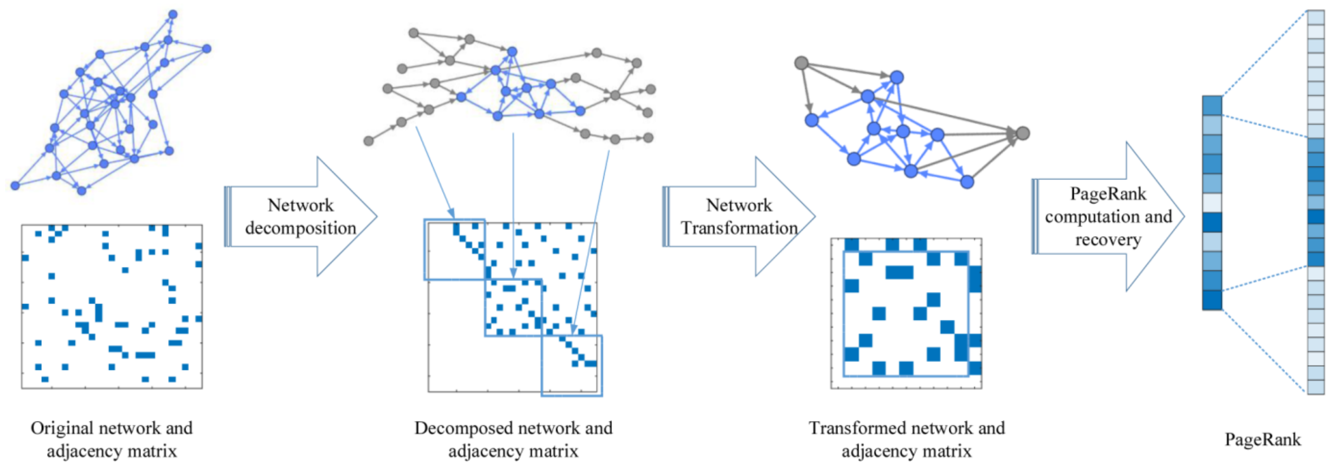
Network decomposition
Network Transformation
PageRank computation and recovery
Original network and adjacency matrix
Decomposed network and adjacency matrix
Transformed network and adjacency matrix
PageRank

FIGURE 1. The flowchart of Fast PageRank. The whole process is divided into network decomposition, network transformation, and PageRank computation and recovery. The original network is decomposed into three parts, including two sub-DAGs and one general sub-network. Then, two sub-DAGs are lumped into two nodes respectively based on a similarity transformation, such that the original network is transformed into a much smaller one. Finally, PageRank on this small network is computed and used to recover the original PageRank.

how to transform the original PageRank problem into a smaller one, and recover the result of the original problem. Finally, the algorithm analysis is provided.

A. NETWORK DECOMPOSITION

Same as section II, we denote a general directed network as $G = (V, E)$. For $\forall i, j \in V$, if there is a path that leads from i to j, we call that j is reachable from i. Specifically, if one node is reachable from itself, a cycle exists in the network. Now, we introduce a particular technique to decompose the original network into three parts, including two sub-DAGs and one general sub-network.

As aforementioned, the nodes without out-links are dangling nodes in a network, and as [26] states, the nodes without in-links are referred to as unreferenced nodes. In the adjacency matrix A, the rows corresponding to dangling nodes are $\mathbf{0}^T$ rows, and the columns corresponding to unreferenced nodes are $\mathbf{0}$ columns. Moreover, if some nodes are isolated, namely both dangling nodes and unreferenced nodes, we regard them as the dangling nodes to simplify the description. By permuting the rows and columns of A, we can achieve the goal that all $\mathbf{0}^T$ rows are at the bottom of A and all $\mathbf{0}$ columns are at the left of A, namely

$$A = \begin{bmatrix} 0 & * & * \\ 0 & \bar{A} & * \\ 0 & 0 & 0 \end{bmatrix} \tag{6}$$

where $\bar{A}$ represents the links among nodes that are neither dangling nodes nor unreferenced nodes.

The matrix $\bar{A}$ can be regarded as an adjacency matrix of a sub-network in G, thus there may exist $\mathbf{0}^T$ rows and $\mathbf{0}$ columns, that is, the sub-network also contains dangling nodes and unreferenced nodes. Likewise, we permute $\bar{A}$ with the above strategy that setting all $\mathbf{0}^T$ rows at bottom and all $\mathbf{0}$ columns at left, namely we separate the dangling nodes and unreferenced nodes of the sub-network corresponding to $\bar{A}$. This process of locating zero rows and columns can be repeated recursively on the smaller and smaller middle sub-matrices, until there are no zero rows and columns. Eventually, the adjacency matrix A is rearranged as below,

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ 0 & A_{22} & A_{23} \\ 0 & 0 & A_{33} \end{bmatrix} \tag{7}$$

where A_{11} and A_{33} are strictly upper triangle matrices.

Finally, all nodes in the network G are sorted into three classes: n_1 nodes corresponding to A_{11} are the unreferenced nodes in all middle sub-matrices, denoted as general unreferenced nodes; n_2 nodes corresponding to A_{22} are the core nodes; and n_3 nodes corresponding to A_{33} are general dangling nodes containing the dangling nodes in all middle sub-matrices ($n_1 + n_2 + n_3 = n$). Due to the above recursive permutation, both A_{11} and A_{33} are strictly upper triangle matrices, and their corresponding sub-networks are DAGs. Therefore, the above process decomposes the original network into two sub-DAGs and one general sub-network. Moreover, this decomposition is optimal, since there are no dangling nodes and unreferenced nodes in the sub-network corresponding to A_{22}. The transition matrix P of (7) has following structure,

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ 0 & P_{22} & P_{23} \\ 0 & 0 & P_{33} \end{bmatrix} \tag{8}$$

where P_{11} and P_{33} are also two strictly upper triangle matrices.

Algorithm 1 depicts our network decomposition algorithm in detail. Inspired by the citation network where each article has an attribute as the published time, we artificially assign a time attribute t_i for each node to observe the possible link relations among nodes. Our algorithm searches all edges related to general unreferenced nodes and general dangling nodes, which is much smaller than $|E|$. In the algorithm, r_i is an indicator counting the total times that node i has been visited. When r_i is equal to the in/out-degree of a general unreferenced/dangling node i, all its in/out-links have been visited and t_i would not be updated later. The algorithm ensures that for $\forall i, j \in V_u$, $t_i < t_j$ if $(i,j) \in E$, and for $\forall i, j \in V_d$, $t_i > t_j$ if $(i,j) \in E$. Then we can compare any pair of nodes by their time attributes, which extends V_u and V_d to the totally ordered sets. After sorting nodes in V_u in ascending order and nodes in V_d in descending order by their time attributes, the adjacency matrix is formed as (7).

Algorithm 1 Network Decomposition

```
Input: $V$, the set of nodes; $E$, the set of edges.
Output: $V_u$, general unreferenced nodes;
        $V_c$, center nodes;
        $V_d$, general dangling nodes.
 1) $V_u = V_d = \Phi$
 2) for $\forall i \in V$
 3)     set $t_i = 1, r_i = 0$
 4)     if $|B_i| = 0$ add $i$ to list $L_u$, else if $|F_i| = 0$ add $i$ to
          list $L_d$
 5) for node $i$ in $L_u$
 6)     add $i$ to $V_u$
 7)     for $j \in F_i$: $r_j = r_j + 1$, $t_j = \max\{t_j, t_i + 1\}$,
          if $r_j = |B_j|$, add $j$ to $L_{next}$
 8)     if $L_{next}$ is not empty, set $L_u = L_{next}$ and clear $L_{next}$,
          else break
 9) for node $i$ in $L_d$
10)     add $i$ to $V_d$
11)     for $j \in B_i$: $r_j = r_j + 1$, $t_j = \max\{t_j, t_i + 1\}$,
          if $r_j = |F_j|$, add $j$ to $L_{next}$
12)     if $L_{next}$ is not empty, set $L_d = L_{next}$ and clear $L_{next}$,
          else break
13) $V_c = V / V_u / V_d$
14) return $V_u, V_c, V_d$.
```

B. FAST PAGERANK COMPUTATION

Based on the above network decomposition, we introduce a method to accelerate the PageRank computation. Our solution allows to transform the original PageRank problem into a PageRank problem on a much smaller network, then recover the results for the original problem. It minimizes the number of nodes whose PageRank values need to be iteratively computed. In order to achieve our goal, we first show that the PageRank problem introduced in section II is equivalent to a corresponding linear system [27], [28].

Theorem 1: Given a PageRank problem with the damping factor d, the teleportation vector v, and the initial transition matrix P. The estimation of PageRank by the eigenvector problem $p^{\mathrm{T}}P'' = p^{\mathrm{T}}$ with $p^{\mathrm{T}}e = 1$ is equivalent to the linear system $x^{\mathrm{T}}(I - dP) = u^{\mathrm{T}}$ with $v = u/u^{\mathrm{T}}e$, and normalizing x to sum to 1, that is $p = x/x^{\mathrm{T}}\ e$.

Proof: According to (5), $p^{\mathrm{T}} = p^{\mathrm{T}}P'' = dp^{\mathrm{T}}(P + cv^{\mathrm{T}}) + (1-d)v^{\mathrm{T}}$, thus we have

$$\begin{aligned} p^{\mathrm{T}}(I - dP) &= dp^{\mathrm{T}}cv^{\mathrm{T}} + (1-d)v^{\mathrm{T}} \\ &= p^{\mathrm{T}}(dc + (1-d)e)v^{\mathrm{T}} \end{aligned}$$

Regarding the linear system $x^{\mathrm{T}}(I - dP) = u^{\mathrm{T}} = (u^{\mathrm{T}}e)v^{\mathrm{T}}$, if $p = x/x^{\mathrm{T}}e$, then $p^{\mathrm{T}}(I - dP) = (u^{\mathrm{T}}e/x^{\mathrm{T}}e)v^{\mathrm{T}}$. Therefore, we only need to prove that $p^{\mathrm{T}}(dc + (1-d)e) = u^{\mathrm{T}}e/x^{\mathrm{T}}e$. Since $c = [c_1, \ldots, c_n]^{\mathrm{T}}$ indicates which nodes are the dangling nodes, hence $c = e - Pe$. Under condition $p = x/x^{\mathrm{T}}e$, we have

$$\begin{aligned} x^{\mathrm{T}}ep^{\mathrm{T}}(dc + (1-d)e) &= x^{\mathrm{T}}(d(I-P)e + (1-d)e) \\ &= x^{\mathrm{T}}(I - dP)e = u^{\mathrm{T}}e \end{aligned}$$

The equation has been proved. □

Imagining the PageRank values as traffics flowing along the edges in a network [40], they will keep a stationary state when the incoming traffic for each node is equal to its out traffic, then the PageRank values are the steady traffics. Regarding the DAG in Fig.2, the ancestor nodes of any node cannot overlap with its descendent nodes, and the traffic of one node can only flow to its descendant nodes without flowing circularly. That is, the PageRank value of one node only depends on its ancestor nodes in DAGs. This property is beneficial to lump two sub-DAGs corresponding to P_{11} and P_{33} in (8) into two single nodes, such that reducing the network size of PageRank problem.

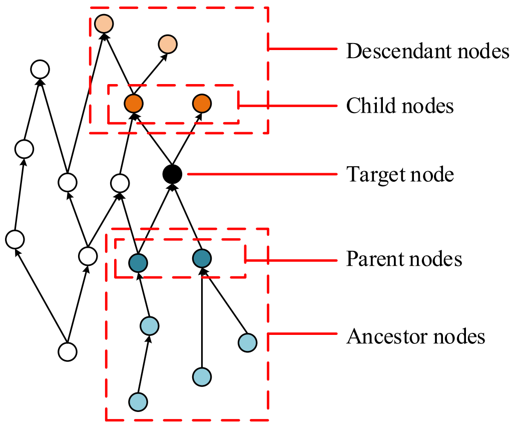

FIGURE 2. An example of directed acyclic graph.

Partitioning $v^{\mathrm{T}} = [v_1^{\mathrm{T}}, v_2^{\mathrm{T}}, v_3^{\mathrm{T}}]$ with v_1^{T} being $n_1 \times 1$, v_2^{T} being $n_2 \times 1$, and v_3^{T} being $n_3 \times 1$, consistently with the network decomposition result, we define a matrix of order $n_2 + 2$,

$$\bar{P} = \begin{bmatrix} u_1^{\mathrm{T}}P_{11}e & h^{\mathrm{T}}(P_{11}eu_2^{\mathrm{T}} + P_{12}) & h^{\mathrm{T}}(P_{11}eu_3^{\mathrm{T}} + P_{13})e \\ 0 & P_{22} & P_{23}e \\ 0 & 0 & 0 \end{bmatrix} \quad (9)$$

where $u = [u_1^{\mathrm{T}}, u_2^{\mathrm{T}}, u_3^{\mathrm{T}}] = [v_1^{\mathrm{T}}(I - dP_{11})^{-1}, v_2^{\mathrm{T}}, v_3^{\mathrm{T}}]/[v_1^{\mathrm{T}}(I - dP_{11})^{-1}, v_2^{\mathrm{T}}, v_3^{\mathrm{T}}]e$ and $h^{\mathrm{T}} = u_1^{\mathrm{T}}/u_1^{\mathrm{T}}e$.

The matrix $\bar{P}$ is row-stochastic except for the last $\mathbf{0}^{\mathrm{T}}$ row. Therefore, it can be regarded as an initial transition matrix of a PageRank problem on a network $\bar{G}$ containing $n_2 + 2$ nodes. Comparing P with $\bar{P}$, we observe that the new network $\bar{G}$ is formed by lumping the general unreferenced nodes and general dangling nodes in G, as well as adjusting the weights of corresponding edges. Theorem 2 presents our major conclusion which demonstrates this lumping process via a similarity transformation, and provides an expression for the original PageRank vector p^{T} in terms of the result of a PageRank problem on $\bar{G}$.

Theorem 2: Given a PageRank problem on a network G that containing initial transition matrix P, damping factor d, and teleportation vector v, a new network $\bar{G}$ with n_2+2 nodes can be formed by lumping general unreferenced nodes and general dangling nodes in G via a similarity transformation. Then, if $\pi^{\mathrm{T}} = \left[\pi_1, \pi^{\mathrm{T}}_{2:n_2+1}, \pi_{n_2+2}\right]$, *where π_1 and π_{n_2+2} are two scalars, is the PageRank vector on $\bar{G}$ with damping factor d and teleportation vector u, the original PageRank vector is given by* $p = x/x^{\mathrm{T}}e$ *with* $x^{\mathrm{T}} = [x_1^{\mathrm{T}}, x_2^{\mathrm{T}}, x_3^{\mathrm{T}}]$, *where*

$$\begin{aligned} x_1^{\mathrm{T}} &= \pi_1 h^{\mathrm{T}} \\ x_2^{\mathrm{T}} &= \pi^{\mathrm{T}}_{2:n_2+1} \\ x_3^{\mathrm{T}} &= ((1-d+d\pi_{n_2+2}+dx_1^{\mathrm{T}}P_{11}e)u_3^{\mathrm{T}} \\ &\quad + dx_1^{\mathrm{T}}P_{13} + dx_2^{\mathrm{T}}P_{23})(I-dP_{33})^{-1} \end{aligned} \tag{10}$$

Proof: According to the network decomposition result (8) and Theorem 1, the original PageRank problem is equivalent to the linear system $x^{\mathrm{T}}(I - dP) = v^{\mathrm{T}}$, namely

$$[x_1^{\mathrm{T}}, x_2^{\mathrm{T}}, x_3^{\mathrm{T}}]\begin{bmatrix} I-dP_{11} & -dP_{12} & -dP_{13} \\ 0 & I-dP_{22} & -dP_{23} \\ 0 & 0 & I-dP_{33} \end{bmatrix} = [v_1^{\mathrm{T}}, v_2^{\mathrm{T}}, v_3^{\mathrm{T}}]$$

Let $y^{\mathrm{T}} = [y_1^{\mathrm{T}}, y_2^{\mathrm{T}}, y_3^{\mathrm{T}}] = [x_1^{\mathrm{T}}, x_2^{\mathrm{T}}, x_3^{\mathrm{T}}(I - dP_{33})]$ and $w^{\mathrm{T}} = [w_1^{\mathrm{T}}, w_2^{\mathrm{T}}, w_3^{\mathrm{T}}] = [v_1^{\mathrm{T}}(I - dP_{11})^{-1}, v_2^{\mathrm{T}}, v_3^{\mathrm{T}}]$, we have $y^{\mathrm{T}}(I - dQ) = w^{\mathrm{T}}$, where

$$Q = \begin{bmatrix} 0 & P_{12} & P_{13} \\ 0 & P_{22} & P_{23} \\ 0 & 0 & 0 \end{bmatrix}$$

Based on Theorem 1, this linear system is equivalent to a PageRank problem with the damping factor d, the teleportation vector $u = [u_1^{\mathrm{T}}, u_2^{\mathrm{T}}, u_3^{\mathrm{T}}] = w/w^{\mathrm{T}}e$, and initial transition matrix $Q + bu^{\mathrm{T}}$ where $b = e - Qe$. Assuming that σ^{T} is the result of this PageRank problem, thus we have $\sigma^{\mathrm{T}}Q'' = \sigma^{\mathrm{T}}$ with

$$\begin{aligned} Q'' &= d(Q + bu^{\mathrm{T}}) + (1-d)eu^{\mathrm{T}} \\ &= \begin{bmatrix} Q''_{11} & Q''_{12} & Q''_{13} \\ (1-d)eu_1^{\mathrm{T}} & Q''_{22} & Q''_{23} \\ eu_1^{\mathrm{T}} & eu_2^{\mathrm{T}} & eu_3^{\mathrm{T}} \end{bmatrix} \end{aligned}$$

where

$$\begin{aligned} Q''_{11} &= (1-d)eu_1^{\mathrm{T}} + dP_{11}eu_1^{\mathrm{T}} \\ Q''_{12} &= (1-d)eu_2^{\mathrm{T}} + dP_{11}eu_2^{\mathrm{T}} + dP_{12} \\ Q''_{13} &= (1-d)eu_3^{\mathrm{T}} + dP_{11}eu_3^{\mathrm{T}} + dP_{13} \\ Q''_{22} &= (1-d)eu_2^{\mathrm{T}} + dP_{22} \\ Q''_{23} &= (1-d)eu_3^{\mathrm{T}} + dP_{23} \end{aligned}$$

Define a similarity transformation matrix S by

$$S = \begin{bmatrix} L_1^{-1} & 0 & 0 \\ 0 & I_{n_2} & 0 \\ 0 & 0 & L_2 \end{bmatrix}$$

where $L_2 = I_{n_3} - \hat{e}e^{\mathrm{T}}/n_3$ with $\hat{e} = e - e_1 = [0, 1, 1, \ldots, 1]^{\mathrm{T}}$, and $L_1 = \begin{bmatrix} l^{\mathrm{T}} & 1 \\ -I_{n_1-1} & e \end{bmatrix}$ with $l^{\mathrm{T}} = [u_{1,2}/u_{1,1}, u_{1,3}/u_{1,1}, \cdots, u_{1,n_1}/u_{1,1}]$ without loss of generality. Their inverse matrices are $L_1^{-1} = \frac{1}{l^{\mathrm{T}}e+1}\begin{bmatrix} e & el^{\mathrm{T}} - (l^{\mathrm{T}}e+1)I \\ 1 & l^{\mathrm{T}} \end{bmatrix}$ and $L_2^{-1} = I_{n_3} + \hat{e}e^{\mathrm{T}}$. Then we achieve the similarity transformation for Q'' by S. Set $h^{\mathrm{T}} = [1, l^{\mathrm{T}}]/(l^{\mathrm{T}}e + 1) = u_1^{\mathrm{T}}/u_1^{\mathrm{T}}e$, it follows that

$$SQ''S^{-1} = \begin{bmatrix} 0 & * & * \\ 0 & Q_1 & Q_2 \\ 0 & 0 & 0 \end{bmatrix}$$

where

$$Q_1 = \begin{bmatrix} h^{\mathrm{T}}Q''_{11}e & h^{\mathrm{T}}Q''_{12} & h^{\mathrm{T}}Q''_{13}e \\ (1-d)eu_1^{\mathrm{T}}e & Q''_{22} & Q''_{23}e \\ u_1^{\mathrm{T}}e & u_2^{\mathrm{T}} & u_3^{\mathrm{T}}e \end{bmatrix}$$

$$\begin{aligned} Q_2 &= \begin{bmatrix} h^{\mathrm{T}}Q''_{13} \\ Q''_{23} \\ u_3^{\mathrm{T}} \end{bmatrix} L_2^{-1}[e_2 \cdots e_{n_3}] \\ &= \begin{bmatrix} (1-d+dh^{\mathrm{T}}P_{11}e)u_3^{\mathrm{T}} + dh^{\mathrm{T}}P_{13}) \\ (1-d)eu_3^{\mathrm{T}} + dP_{23} \\ u_3^{\mathrm{T}} \end{bmatrix} \\ &\quad \times (I_{n_3}^{\mathrm{T}} + \hat{e}e^{\mathrm{T}})[e_2 \cdots e_{n_3}] \end{aligned}$$

Due to the similarity transformation, the matrix Q_1 is row-stochastic of order $n_2 + 2$ with the same nonzero eigenvalues as Q''. Further, Q_1 is a modified transition matrix of the PageRank problem with u^{T}, d, and initial transition matrix (9)

$$\bar{P} = \begin{bmatrix} u_1^{\mathrm{T}}P_{11}e & h^{\mathrm{T}}(P_{11}eu_2^{\mathrm{T}} + P_{12}) & h^{\mathrm{T}}(P_{11}eu_3^{\mathrm{T}} + P_{13})e \\ 0 & P_{22} & P_{23}e \\ 0 & 0 & 0 \end{bmatrix}$$

Finally, we transform the original PageRank problem on a network G with n nodes into the problem on a smaller network $\bar{G}$ with $n_2 + 2$ nodes.

In the following, we show how to derive the original PageRank vector by this smaller problem. Assume π is the PageRank vector of $\bar{P}$ with u^{T} and d, namely $\pi^{\mathrm{T}}Q_1 = \pi^{\mathrm{T}}$, then the vector $[0, \pi^{\mathrm{T}}, \pi^{\mathrm{T}}Q_2]$ is an eigenvector for $SQ''S^{-1}$

associated with the eigenvalue 1. Due to the similarity transformation, Q'' has the same nonzero eigenvalues as Q_1. Hence, $[0, \pi^T, \pi^T Q_2]S$ is an eigenvector of Q'' with the eigenvalue 1, and a multiple of the stationary distribution σ^T of Q''. By the following derivation, $[0, \pi^T, \pi^T Q_2]Se=1$, thus we express $\sigma^T = [0, \pi^T, \pi^T Q_2]S$. Return to the original partition as (7) which separates the leading n_1 elements and the last n_3 elements,

$$\begin{aligned}\sigma^T &= \left[\sigma_1^T, \sigma_2^T, \sigma_3^T\right] \\ &= \left[(0^T\pi_1),\ \pi^T_{2:n_2+1},\ (\pi_{n_2+2}\pi^T Q_2)\right]\begin{bmatrix} L_1^{-1} & 0 & 0 \\ 0 & I_{n_2} & 0 \\ 0 & 0 & L_2\end{bmatrix} \\ &= \left[(0^T\pi_1)L_1^{-1},\ \pi^T_{2:n_2+1},\ (\pi_{n_2+2}\pi^T Q_2)L_2\right] \\ &= \left[\pi_1 h^T,\ \pi^T_{2:n_2+1},\ (\pi_{n_2+2}\pi^T Q_2)L_2\right]\end{aligned}$$

Regarding the trailing n_3 elements, we have

$$\begin{aligned}\sigma_3^T &= (\pi_{n_2+2},\ \pi^T Q_2)L_2 = \pi^T\begin{bmatrix} h^T Q''_{13} \\ Q''_{23} \\ u_3^T\end{bmatrix} \\ &= \pi_1((1-d)u_3^T + dh^T(P_{11}eu_3^T + P_{13})) \\ &\quad + \pi^T_{2:n_2+1}((1-d)eu_3^T + dP_{23}) + \pi_{n_2+2}u_3^T \\ &= (1-d+d\pi_{n_2+2} + d\sigma_1^T P_{11}e)u_3^T + d\sigma_1^T P_{13} + d\sigma_2^T P_{23}\end{aligned}$$

Since y^T is a multiple of σ^T and $x^T = [x_1^T, x_2^T, x_3^T] = [y_1^T, y_2^T, y_3^T(I-dP_{33})^{-1}]$, we have

$$x^T = \left[x_1^T, x_2^T, x_3^T\right] \propto [\sigma_1^T, \sigma_2^T, \sigma_3^T(I-dP_{33})^{-1}]$$

Meanwhile, x^T is a multiple of p^T with $p^T e = 1$, therefore the original PageRank vector is calculated by $p^T = [\sigma_1^T, \sigma_2^T, \sigma_3^T(I-dP_{33})^{-1}]/[\sigma_1^T, \sigma_2^T, \sigma_3^T(I-dP_{33})^{-1}]e$. This completes the proof of the theorem. □

Theorem 2 shows that to compute the PageRank vector p^T of the original network, we can solve a PageRank problem on a much smaller network, and then recover the PageRank vector p^T based on (10). Since our focus is the relative PageRank values rather than the absolute ones, the process for normalization is usually unnecessary. The nonnormalized PageRank vector has good potential to make the PageRank values comparable across networks [41], and is robust to the changes of network structure [42].

Algorithm 2 depicts the fast PageRank computation based on Theorem 2. New teleportation vector u^T and initial transition matrix $\bar{P}$ are first computed based on the network decomposition result. Then our algorithm solves the smaller PageRank problem on $\bar{G}$ by an iterative method. Since the new network $\bar{G}$ is much smaller than the original one G, this process can save a large amount of operations per iteration. Finally, we recover the full PageRank vector. Except the iterative computation on new network, other operations are one-off. Two inverse matrices need to be computed, namely $(I-dP_{11})^{-1}$ and $(I-dP_{33})^{-1}$. Since P_{11} and P_{33} are strictly upper triangle, $I-dP_{11}$ and $I-dP_{33}$ are two upper triangle matrices. Taking $R = I-dP_{11}$ for example, elements in the inverse matrix R^{-1} can be computed directly by backward substitution as follows,

$$R_{ij}^{-1} = \begin{cases} 0 & i > j \\ 1 & i = j \\ -1/\sum_{k=i+1}^{j} R_{ik}R_{kj}^{-1} & i < j \end{cases} \tag{11}$$

Algorithm 2 Fast PageRank Computation

```
Input: $P$; $d$; $v^T$.
Output: the PageRank vector $p^T$.
1) compute $u^T$ and $h^T$;
2) construct matrix $\bar{P}$;
3) choose an initial vector $\pi^T \geq 0$ with $||\pi^T|| = 1$;
4) while not converged,
5)     update $\pi^T$ approximate to the true PageRank
       vector of $\bar{P}$;
6) compute $x^T$ with (10);
7) return $x^T/x^T e$.
```

C. ALGORITHM ANALYSIS

The proposed method consists of two different stages in the whole view. The first one is the network decomposition which separates all general unreferenced nodes and general dangling nodes, such that the adjacency matrix is formed as (7). Algorithm 1 assigns the time attributes to achieve this task. The second one is computing the PageRank vector based on Theorem 2. By lumping two sub-DAGs into two single nodes, we transform the PageRank problem on a smaller network. Our method leads to a large reduction of the iteration cost, and should be more competitive as the size of sub-DAGs increases.

As aforementioned, existing lumping methods also decompose part of nodes from the whole network. Compared with our method, earlier mentioned methods only focus on dangling nodes or weakly dangling nodes, and just one refers to unreferenced nodes [26]. With the network decomposition, our method takes all general dangling nodes and general unreferenced nodes into account, to decompose the DAG structures whose PageRank values can be directly computed. It is the extension of existing lumping methods, and much more efficient by reducing the original PageRank problem to a much smaller one.

To a certain extent, our method is also similar to the adaptive method for PageRank computation. The adaptive method indicates that the convergence rates of PageRank values have a nonuniform distribution, that is, some nodes converge to their true PageRank quickly, whereas others take a longer time to converge [19]. In our method, the PageRank values of general dangling nodes and general unreferenced nodes are recovered directly by the non-iterative process which are regarded as the fastest convergence, and others converge to their truth values by the iterative computation.

In the following, we analyze the time complexity of our method. We denote the number of nonzeros in a matrix P as $nnz(P)$, which is also the number of edges in its corresponding network. The first stage is the network decomposition via algorithm 1. It only needs to traverse all edges related to general unreferenced nodes and general dangling nodes. Hence it needs $nnz(P) - nnz(P_{22})$ operations. Then, the PageRank is computed by algorithm 2, containing the iterative computation on a smaller network (step 3-5) and some one-off computations for transformation and recovery (step 1, 2, and 6). The cost of these one-off processes is about $nnz(P) - nnz(P_{22})$ operations, mainly depending on the number of nonzeros in $P_{11}, P_{12}, P_{13}, P_{23}$, and P_{33}. Meanwhile, the PageRank solve on the smaller network requires $O(nnz(P_{22}))$ operations with the power method [27]. Since the time cost of one-off steps is much smaller, the time complexity of fast PageRank computation is about $O(nnz(P_{22}))$. Regarding the PageRank solve on the original network, it needs about $O(nnz(P))$ operations by the power method. Therefore, our method is much more timesaving and the speedup is nearly $nnz(P)/nnz(P_{22})$.

IV. EXPERIMENTS

We perform experiments on different datasets to evaluate the efficiency of fast PageRank computation. As the extension of lumping methods, our method is compared with existing lumping methods firstly. Then, we compare our method with other state-of-the-art methods for further elaboration. Finally, we show an application on citation network. Without loss of generality, we use the uniform vector v, and the convergence tolerance is set to 10^{-10}. Each method is operated 5 times on each dataset, and the average results are shown in the paper. All experiments are conducted on a HP computer with Intel(R) Xeon(R) processer with CPU 3.30GHz, and RAM 16GB, under the Windows 10 64 bit operating system.

A. COMPARISON WITH LUMPING METHOD

Our method and baselines are implemented with $d = 0.85$ on three web graphs, including wb-cs-Stanford matrix, Stanford matrix, and Wikipedia-20070206 matrix. These datasets are publicly available from the university of Florida sparse matrix collection [43]. For each matrix, we set the diagonal elements to be zero, and the final matrices are listed in Table 1. Regarding our method, we use the power method to solve the smaller PageRank problem on $\bar{G}$, denoted as Fast. Three existing lumping methods are implemented as contrast methods, including Lump2, Reorder, and Lump5. Meanwhile, two typical methods (Power and Jacobi) are also applied on the original network. Table 2 summarizes these six methods.

The evaluation metric is the computation time. For Power and Jacobi, the PageRank is computed only by the iterative computation on the whole matrix. Regarding other methods, there are two stages of the PageRank estimation. The first stage is the node reordering or network decomposition, and its time cost is t_1. The second stage is computing the PageRank values containing the iterative computation for the smaller system, and the corresponding time is t_2. Finally the total time cost is $t = t_1 + t_2$ seconds.

We first try to illustrate the reordering strategies or network decomposition in terms of Lump2, Reorder, Lump5, and Fast. Taking the wb-cs-Stanford matrix for example, Fig.3 shows the structures of the original matrix and those

TABLE 1. Experimental web matrices.

Dataset	Wb-cs-Stanford	Stanford	Wikipedia-20070206
Nodes	9914	281903	3566907
Edges	35555	2312497	45013315
Average out-degree	3.59	8.20	12.62
General unreferenced nodes	986	788	1146818
General dangling nodes	2822	24017	48415

TABLE 2. Summary of methods in first experiment.

Abbreviation	Algorithm Description
Power	The power method to compute the PageRank vector [11].
Jacobi	The Jacobi method for solving the original problem [44].
Lump2	Lumping algorithm with respect to 2 types of nodes [24].
Reorder	The recursively reordering algorithm for PageRank with dangling nodes [27].
Lump5	Lumping algorithm with respect to 5 types of nodes [26].
Fast	Fast PageRank computation combined with the power method.

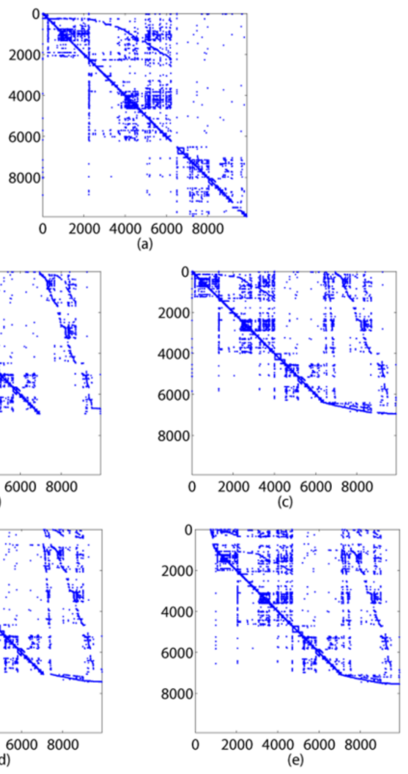

FIGURE 3. Structures of the original and reordered wb-cs-Stanford matrices where the blue nodes are the nonzero elements and white represents the zero elements.

being reordered by these four methods. We observe that much more nodes whose PageRank values can be directly computed, are decomposed from the original matrix (the blank in the left and bottom in each figure). Indeed, the size of the original matrix is 9914, hence Power and Jacobi have to solve a system of size 9914×9914. If the matrix is lumped based on Lump2, one needs to iteratively solve a 6951×6951 system for PageRank. Similarly, Reorder needs to iteratively solve a 6391×6391 system, and Lump5 needs to iteratively solve a 6341×6341 system. As the extension of these methods, Fast computes the result by solving a PageRank problem on a much smaller network with 6108 nodes, which can significantly reduce the iteration time.

Table 3 shows the experimental results of these six methods on each dataset, including the number of iterations and time cost of each method. The best result for each dataset is in bold. Comparing Power with Jacobi, we find that the convergence rates are different for these two methods. The power method is superior to the Jacobi method for wb-cs-Stanford and Wikipedia-20070206 matrices, whereas the Jacobi method is better on the Stanford matrix.

TABLE 3. Results in terms of relevant computation time in first experiment (seconds).

		Wb-cs-Stanford	Stanford	Wikipedia-20070206
Power	iterations	93	104	81
	t	3.518	124.499	1265.141
Jacobi	iterations	105	100	103
	t	3.890	115.245	1464.079
Lump2	iterations	106	100	103
	t_1	0.009	0.223	3.156
	t_2	2.909	105.419	1438.524
	t	2.918	105.642	1441.680
Reorder	iterations	106	100	103
	t_1	0.024	0.418	4.364
	t_2	2.691	101.984	1454.039
	t	2.715	102.402	1458.403
Lump5	iterations	106	100	103
	t_1	0.022	0.434	13.437
	t_2	2.818	104.809	1073.257
	t	2.840	105.243	1086.694
Fast	iterations	93	104	81
	t_1	0.038	0.465	17.756
	t_2	2.302	101.618	887.539
	t	**2.340**	**102.083**	**905.295**

Then we analyze the results of Lump2, Reorder, Lump5 and Fast, since they have similar procedures. According to t_2, the iteration cost of Fast is the smallest, since the problem it solves is much smaller than others. Meanwhile, these four methods have better performance than Jacobi, although they need to reorder nodes firstly. For wb-cs-Stanford and Wikipedia-20070206 matrices, our method has the best performance with 2.340s and 905.295s respectively, against the second best method by a large margin. Regarding Stanford matrix, our method and Reorder have almost the same time cost. The reason is that, there are less general unreferenced nodes in this network, and the cost required by the reordering step may offset the decrease in the size of problem solved by iterative computation.

Finally, we observe that our method is better than the power method. Due to the great reduction of the network size of PageRank problem, the time cost per iteration has been reduced significantly. Hence, although our method consists of two stages, the time for iterative computation is reduced too much such that the time for other one-off steps can be ignored. The speedup between Fast and Power is calculated by $s = t_p/t_f$, where t_p and t_f are the total time cost of Power and Fast. The results for three matrices are 1.50, 1.22 and 1.40 respectively.

B. FURTHER COMPARISION

Our method is a framework for fast PageRank computation, which transforms the original PageRank problem into a much smaller one. In fact, this smaller PageRank problem has flexible solutions, and many advanced methods can be applied to reduce the iteration cost, besides the power method used in the above part. For further elaboration, we conduct experiments on the same datasets in Table 1, by combining our framework with other state-of-the-art methods including Extra, PET, In-Out, PIO, and MPIO. Meanwhile, we also implement these five methods on the whole matrices as baselines. Table 4 illustrates the experimental methods.

TABLE 4. Summary of methods in second experiment.

Abbreviation	Algorithm Description
Extra	The extrapolation method for PageRank computation [18].
PET	The power method with the extrapolation process based on trace [45].
In-Out	The inner-outer stationary method for PageRank [22].
PIO	A two-step matrix splitting iteration method [23].
MPIO	Multi-step power-inner-outer computation method [46].
Fast+Extra	Fast PageRank framework combined with Extra.
Fast+PET	Fast PageRank framework combined with PET.
Fast+In-Out	Fast PageRank framework combined with In-Out.
Fast+PIO	Fast PageRank framework combined with PIO.
Fast+MPIO	Fast PageRank framework combined with MPIO.

Table 5 lists the total computation time of these methods. The best result for each dataset is in bold. Comparing with the methods directly solving problems on the whole matri-

TABLE 5. Time cost of ten methods in second experiment (seconds).

	Wb-cs-Stanford	Stanford	Wikipedia-20070206
Extra	3.443	122.912	1203.984
Fast+Extra	2.342	96.370	827.829
PET	3.148	107.916	1161.729
Fast+PET	2.650	102.457	912.823
In-Out	2.834	111.133	1195.987
Fast+In-Out	1.848	103.327	1023.113
PIO	1.452	65.344	867.805
Fast+PIO	0.974	61.690	654.361
MPIO	0.737	36.593	542.441
Fast+MPIO	**0.526**	**34.292**	**446.630**

ces, we observe that methods combined with fast PageRank computation have significantly better performance. Taking MPIO for example, there are critical improvements for wb-cs-Stanford and Wikipedia-20070206 after being combined with our framework. Even for the Stanford matrix, Fast+MPIO can also be of much less time-consuming than MPIO. Moreover, since MPIO is superior to the Power method, the Jacobi method, Extra, PET, In-Out, and PIO, Fast+MPIO obtains the best results among all methods in Table 3 and 5. Compared with the power method, it provides the speedup of about a factor of 6.69, 3.63 and 2.83 for three matrices respectively.

One point to be aware of is that, although PET outperforms Extra in Table 5, Fast+PET is not better than Fast+Extra. The reason is related to PET method, which is designed based on the condition that copious dangling nodes exist in a network [45]. Since there are many dangling nodes in the original networks, PET is superior to Extra on the whole matrices, which is in accordance with [45]. However, when we combine PET with fast PageRank, our method transforms the original PageRank problem into a new one on a smaller network firstly. This new network contains only one dangling node, which invalidates PET. Therefore, Fast+PET becomes inferior to Fast+Extra. In order to avoid the above case, we should take the characteristics of the state-of-the-art method into consideration, when combining it with our proposed framework for the better performance.

The results demonstrate that our method outperforms baselines as the extension of lumping methods. It is powerful for the PageRank computation by transforming the original problem into a smaller one. Furthermore, fast PageRank can be combined with other advanced methods for further acceleration, and is more competitive for the network whose ratio between $nnz(P)$ and $nnz(P_{22})$ is much bigger.

C. APPLICATION ON CITATION NETWORK

Our method is able to contribute to the advancement of many applications, such as the citation analysis in Academia. The real paper citation network is nearly acyclic with several cycles [47]. To show the applicability of our method, we collect the data during the period January 1988 to April 2014 from the Physical Review Journals. The raw data containing 387977 papers are downloaded from the Web of Science TM Core Collection in May 2014 and trimmed by following preprocessing. We remove all papers that contain attributes of false format or incomplete content. The citations pointing to papers outside the dataset are also removed, that is, we reserve only citations from Physical Review papers to Physical Review papers. The final paper network contains 349022 nodes together with 3029754 directed edges.

To illustrate the cycles in the paper network, we achieve the network decomposition by algorithm 1. There are 174695 general unreferenced nodes and 53135 general dangling nodes, that is, there exist several cycles in this network. This process requires about 3.424 seconds. Then, we compute PageRank on the citation networks with cycles and without cycles by removing several relevant citations. Comparing these two results, we find that the top-1000 papers of two ranks are completely different. It demonstrates that removing these citations has significant effect on the final rank of papers, especially for the papers with bigger PageRank values. Hence, these citations should be considered during the PageRank computation.

Finally, we compute PageRank for the paper network with different damping factors, and six methods including Power, PIO, MPIO, Fast, Fast+PIO, and Fast+MPIO. Fig.4 shows the total time of each method. In addition to modifying the transition matrix of PageRank to overcome some possible problems, the damping factor also controls the convergence rate of the iterative computation [13]. As the damping factor increases, six methods cost much more time to compute PageRank. Moreover, Fast+MPIO has the better performance than MPIO except when the damping factor is equal to 0.15. The reason is MPIO converges so fast when $d = 0.15$, that the cost required by other one-off steps offsets the cost decrease of the iterative computation on the smaller network. With regard to other methods, Fast is superior to Power among all damping factors, and Fast+PIO outperforms PIO as well. Hence, our method can be a good option for the bibliometrics research in citation analysis.

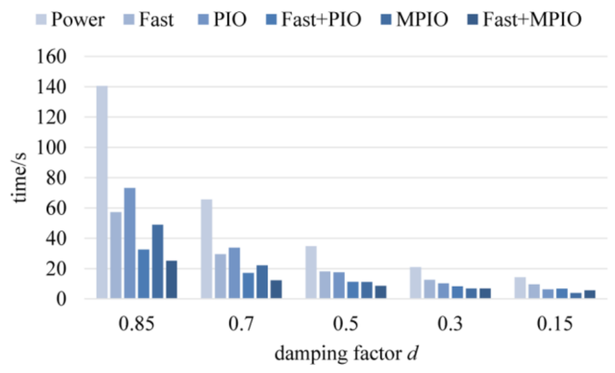

FIGURE 4. Time cost of citation network with different damping factors.

V. CONCLUSION

In this paper, we proposed a fast PageRank computation method, which combines a network decomposition and a transformation of the PageRank problem. According to the decomposition resulting in one general sub-network and two sub-DAGs, we demonstrated that the original PageRank problem can be directly transformed into a smaller one by a similarity transformation. It can be regarded as lumping two sub-DAGs into two single nodes respectively. With attention to all general unreferenced nodes and general dangling nodes, our method transforms the PageRank problem into a new one as small as possible, such that largely reducing the time cost of iterative computation.

Empirical experiments on real datasets are conducted to evaluate the proposed method. The results demonstrate that fast PageRank outperforms the existing alternatives by a large margin, which provides a significant speedup compared with baselines. The efficiency of our method mainly relates to the size of two sub-DAGs, thus it would become more competitive as the size of two sub-DAGs increases. As shown in the case study on citation network, our method allows many applications to be implemented more efficiently. Furthermore, many advanced computation methods can also be embedded in our framework for further improvement.

REFERENCES

[1] Y. Jing and S. Baluja, "VisualRank: Applying PageRank to large-scale image search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1877–1890, Nov. 2008.

[2] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the Web," Dept. Comput. Sci., Stanford Infolab, Stanford, CA, USA, Tech. Rep., 1999. [Online]. Available: http://ilpubs.stanford.edu:8090/422/

[3] U. Senanayake, M. Piraveenan, and A. Zomaya, "The PageRank-index: Going beyond citation counts in quantifying scientific impact of researchers," *PLoS ONE*, vol. 10, no. 8, p. e0134794, 2015.

[4] Z. Li, Q. Peng, and C. Liu, "Two citation-based indicators to measure latent referential value of papers," *Scientometrics*, vol. 108, no. 3, pp. 1299–1313, 2016.

[5] C. Gao, Z. Wang, X. Li, Z. Zhang, and W. Zeng, "PR-index: Using the *h*-index and PageRank for determining true impact," *PLoS ONE*, vol. 11, no. 9, p. e0161755, 2016.

[6] S. J. Yu, "The dynamic competitive recommendation algorithm in social network services," *Inf. Sci.*, vol. 187, pp. 1–14, Mar. 2012.

[7] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM* , vol. 46, no. 5, pp. 604–632, 1999.

[8] J. Bjelland, M. Burgess, G. Canright, and K. Engø-Monsen, "Eigenvectors of directed graphs and importance scores: Dominance, T-Rank, and sink remedies," *Data Mining Knowl. Discovery*, vol. 20, pp. 98–151, Jan. 2010.

[9] H. H. Bi, J. Wang, and D. K. J. Lin, "Comprehensive citation index for research networks," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 8, pp. 1274–1278, Aug. 2011.

[10] J. Liu *et al.*, "A new method to construct co-author networks," *Phys. A, Stat. Mech. Appl.*, vol. 419, pp. 29–39, Feb. 2015.

[11] P. Berkhin, "A survey on PageRank computing," *Internet Math.*, vol. 2, no. 1, pp. 73–120, 2005.

[12] M. Bianchini, M. Gori, and F. Scarselli, "Inside PageRank," *ACM Trans. Internet Technol.*, vol. 5, no. 1, pp. 92–128, 2005.

[13] A. N. Langville and C. D. Meyer, "Deeper inside PageRank," *Internet Math.*, vol. 1, no. 3, pp. 335–380, 2004.

[14] F. Chung, "A brief survey of PageRank algorithms," *IEEE Trans. Netw. Sci. Eng.*, vol. 1, no. 1, pp. 38–42, Jan. 2014.

[15] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin, "PageRank computation and the structure of the Web: Experiments and algorithms," in *Proc. 11th Int. World Wide Web Conf., Poster Track*, 2002, pp. 107–117.

[16] A. Sidi, "Vector extrapolation methods with applications to solution of large systems of equations and to PageRank computations," *Comput. Math. Appl.*, vol. 56, no. 1, pp. 1–24, 2008.

[17] H. Migallón, V. Migallón, J. A. Palomino, and J. Penadés, "A heuristic relaxed extrapolated algorithm for accelerating PageRank," *Adv. Eng. Softw.*, vol. 120, pp. 88–95, Jun. 2018.

[18] T. Haveliwala, A. Kamvar, K. Dan, C. Manning, and G. Golub, "Computing PageRank using power extrapolation," Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep., 2003. [Online]. Available: http://ilpubs.stanford.edu:8090/605/

[19] S. Kamvar, T. Haveliwala, and G. Golub, "Adaptive methods for the computation of PageRank," *Linear Algebra Appl.*, vol. 386, pp. 51–65, Jul. 2004.

[20] H. Ishii and R. Tempo, "Distributed randomized algorithms for the PageRank computation," *IEEE Trans. Autom. Control*, vol. 55, no. 9, pp. 1987–2002, Sep. 2010.

[21] H. Ishii, R. Tempo, and E. W. Bai, "A Web aggregation approach for distributed randomized PageRank algorithms," *IEEE Trans. Autom. Control*, vol. 57, no. 11, pp. 2703–2717, Nov. 2012.

[22] D. F. Gleich, A. P. Gray, C. Greif, and T. Lau, "An inner-outer iteration for computing PageRank," *SIAM J. Sci. Comput.*, vol. 32, no. 1, pp. 349–371, 2010.

[23] C. Q. Gu, F. Xie, and K. Zhang, "A two-step matrix splitting iteration for computing PageRank," *J. Comput. Appl. Math.*, vol. 278, pp. 19–28, Apr. 2015.

[24] I. C. F. Ipsen and T. M. Selee, "PageRank computation, with special attention to dangling nodes," *SIAM J. Matrix Anal. Appl.*, vol. 29, no. 4, pp. 1281–1296, 2007.

[25] Y. Q. Lin, X. H. Shi, and Y. M. Wei, "On computing PageRank via lumping the Google matrix," *J. Comput. Appl. Math.*, vol. 224, no. 2, pp. 702–708, 2009.

[26] Q. Yu, Z. K. Miao, G. Wu, and Y. Wei, "Lumping algorithms for computing Google's PageRank and its derivative, with attention to unreferenced nodes," *Inf. Retr.*, vol. 15, no. 6, pp. 503–526, 2012.

[27] A. N. Langville and C. D. Meyer, "A reordering for the PageRank problem," *SIAM J. Sci. Comput.*, vol. 27, no. 6, pp. 2112–2120, 2006.

[28] Y.-M. Bu and T.-Z. Huang, "An adaptive reordered method for computing PageRank," *J. Appl. Math.*, vol. 2013, Jul. 2013, Art. no. 507915.

[29] S. Allesina and M. Pascual, "Googling food Webs: Can an eigenvector measure species' importance for coextinctions," *PLoS Comput Biol*, vol. 5, no. 9, p. e1000494, 2009.

[30] D. Wang, J. A. Wang, M. Lu, F. Song, and Q. Cui, "Inferring the human microRNA functional similarity and functional network based on microRNA-associated diseases," *Bioinformatics*, vol. 26, no. 13, pp. 1644–1650, 2010.

[31] P. J. Ding, J. W. Luo, Q. Xiao, and X. T. Chen, "A path-based measurement for human miRNA functional similarities using miRNA-disease associations," *Sci. Rep.*, vol. 6, p. 32533, Sep. 2016.

[32] S. Fortunato and A. Flammini, "Random walks on directed networks: The case of PageRank," *Int. J. Bifurcation Chaos*, vol. 17, pp. 2343–2353, Jul. 2007.

[33] T. H. Haveliwala, "Topic-sensitive PageRank: A context-sensitive ranking algorithm for Web search," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 4, pp. 784–796, Jul. 2003.

[34] J. H. Kim, K. S. Candan, and M. L. Sapino, "Locality-sensitive and re-use promoting personalized PageRank computations," *Knowl. Inf. Syst.*, vol. 47, no. 2, pp. 261–299, 2016.

[35] M. Pirouz and J. Zhan, "Toward efficient hub-less real time personalized PageRank," *IEEE Access*, vol. 5, pp. 26364–26375, 2017.

[36] G. Iván 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, 2011.

[37] M. Shrestha and C. Moore, "Message-passing approach for threshold models of behavior in networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 89, no. 2, p. 022805, 2014.

[38] J. Heidemann, M. Klier, and F. Probst, "Identifying key users in online social networks: A PageRank based approach," in *Proc. ICIS*, 2010, pp. 1–21.

[39] K. Weiand, F. Kneißl, W. Łobacz, T. Furche, and F. Bry, "PEST: Fast approximate keyword search in semantic data using eigenvector-based term propagation," *Inf. Syst.*, vol. 37, no. 4, pp. 372–390, 2012.

[40] X. H. Li, J. Kurths, C. Gao, J. W. Zhang, Z. Wang, and Z. L. Zhang, "A hybrid algorithm for estimating origin-destination flows," *IEEE Access*, vol. 6, 2018.

[41] K. Berberich, S. Bedathur, G. Weikum, and M. Vazirgiannis, "Comparing apples and oranges: Normalized PageRank for evolving graphs," in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 1145–1146.

[42] C. Engström and S. Silvestrov. (2013). "PageRank for evolving link structures." [Online]. Available: https://arxiv.org/abs/1401.6092

[43] T. A. Davis and Y. F. Hu, "The University of Florida sparse matrix collection," *ACM Trans. Math. Softw.*, vol. 38, no. 1, 2011, Art. no. 1.

[44] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*. Philadelphia, PA, USA: SIAM, 2000.

[45] X. Y. Tan, "A new extrapolation method for PageRank computations," *J. Comput. Appl. Math.*, vol. 313, pp. 383–392, Mar. 2017.

[46] C. Wen, T.-Z. Huang, and Z.-L. Shen, "A note on the two-step matrix splitting iteration for computing PageRank," *J. Comput. Appl. Math.*, vol. 315, pp. 87–97, May 2017.

[47] J. D. West, I. Wesley-Smith, and C. T. Bergstrom, "A recommendation system based on hierarchical clustering of an article-level citation network," *IEEE Trans. Big Data*, vol. 2, no. 2, pp. 113–123, Jun. 2016.

ZHIBO ZHU received the B.Sc. degree in automation science and technology from Xi'an Jiaotong University, China, in 2013, where he is currently pursuing the Ph.D. degree with the Systems Engineering Institute. His current research interests include time series analysis, network analysis, and deep learning for these areas.

QINKE PENG received the B.Sc. degree in mathematics and the M.Sc. and Ph.D. degrees in systems engineering from Xi'an Jiaotong University, China, in 1983, 1986, and 1990, respectively. He was with the CIMS Research Center, Xi'an Jiaotong University, before 1994. He is currently the Director of the Department of Automation Science and Technology and a Faculty Member with the Systems Engineering Institute, Xi'an Jiaotong University. His research interests focus on mining, modeling, and analysis of big data with the application of online public opinion monitoring, biological, network, and financial information analysis, and so on.

ZHI LI received the B.Sc. degree in automation science and technology and the M.Sc. degree in systems engineering from Xi'an Jiaotong University, China, in 2013 and 2016, respectively. His M.Sc. research area was citation network analysis and recommendation algorithm for the literature. He is currently a Manager Candidate with Works Applications China Company Ltd.

XINYU GUAN received the B.Sc. degree in automation science and technology from Xi'an Jiaotong University, China, in 2013, where he is currently pursuing the Ph.D. degree with the Systems Engineering Institute. His major research interests are recommender systems and natural language processing.

OWAIS MUHAMMAD received the B.Sc. degree in electronics engineering from Dawood-UET Karachi, Pakistan, in 2015. He is currently pursuing the M.Sc. degree with the Systems Engineering Institute, Xi'an Jiaotong University, China. His research interests include network data analysis and deep learning.