Graph Ranking Auditing: Problem Definition and Fast Solutions

Meijia Wang, Jian Kang, Nan Cao, Yinglong Xia, Wei Fan, and Hanghang Tong

Abstract—Ranking on graphs is a centerpiece in many high-impact application domains, such as information retrieval, recommender systems, team management, neuroscience and many more. PageRank, along with many of its variants, is widely used across these application domains thanks to its mathematical elegance and the superior performance. Although PageRank and its variants are effective in ranking nodes on graphs, they often lack an efficient and effective way to *audit* the ranking results in terms of the input graph structure, e.g., which node or edge in the graph contributes most to the top-1 ranked node; which subgraph plays a crucial role in generating the overall ranking result? In this paper, we propose to audit graph ranking by finding the influential graph elements (e.g., edges, nodes, attributes, and subgraphs) regarding their impact on the ranking results. First, we formulate graph ranking auditing problem as quantifying the influence of graph elements on the ranking results. Second, we show that our formulation can be applied to a variety of graph structures. Third, we propose effective and efficient algorithms to find the top-k influential edges/nodes/subgraph. Finally, we perform extensive empirical evaluations on real-world datasets to demonstrate that the proposed methods (AURORA) provide intuitive auditing results with linear scalability.

Index Terms—Graph mining, pagerank, explainability

1 INTRODUCTION

THE seminal work of PageRank algorithm [1] has inspired many ranking algorithms on graphs in the past two decades to serve different application purposes. To name a few, Marco and Augusto [2] design ItemRank scoring algorithm to rank products in recommender system; Jianshu *et al.* [3] develop TwitterRank, tailored explicitly for identifying important users in a social network; Rohit *et al.* [4] propose IsoRank for aligning protein-protein interaction network in computational biology. Variants of PageRank algorithms can also be found in other application domains, e.g., neuroscience [5], sports team management [6] and many more. The success of PageRank and its variants in social and economic domains largely relies on their superior ability in identifying important nodes in large-scale graphs.

The popularity of PageRank and its variants hinge on their elegant idea and mathematical simplicity. The mechanism of these ranking algorithms can be viewed as an iterative election process among nodes, where the nodes assign their votes to and gain their votes from others through the edges. Normalization is introduced to ensure that the process will converge and the relative importance of the nodes is determined in terms of the weights they gain from their neighbors. Intuitively, PageRank and its variants closely resemble many phenomena in the real world: celebrities following celebrities in social networks; similar users buying similar items in recommender systems and influential articles inspiring many more important articles in citation networks. Very often, the ranking algorithms can be computed using power iteration method with linear complexity and can further be improved by more efficient algorithms [7], [8].

Successful as it is, interests are naturally drawn to manipulate the ranking results. Hence it is of key importance to identify the vulnerabilities in graph structure against malicious attacks, so as to improve the reliability of the ranking results. To achieve this goal, we need to understand *why* the algorithm outputs a certain ranking result. Instead of knowing the general mathematical mechanism behind these algorithms, it is highly desirable to find out explicitly which element (e.g., edge, node, subgraph, etc.) in a graph dataset makes the most significant contribution to the ranking result. We envision that auditing graph ranking is highly beneficial. It will help identify the vulnerabilities in the graph (e.g., links among three clusters, the bridging node of three clusters in Fig. 1), which will further enable effective defending strategies to avoid malicious manipulation of the ranking results.

In this paper, we aim to audit the ranking by finding the most influential graph elements (e.g., edges, nodes, subgraphs), which we formulate as Graph Ranking Auditing Problem. To be specific, we address two key challenges as follows. First, to quantitatively understand the ranking algorithms, we need an influence measure to assess how the ranking results would change if we perturb a specific graph element. Second, due to its combinatorial nature, effectively

- *M. Wang is with the Arizona State University, Tempe, AZ 85281 USA. E-mail: mwang164@asu.edu.*
- *J. Kang and H. Tong are with the University of Illinois at Urbana-Champaign, Urbana, IL 61820 USA. E-mail: {jiank2, htong}@illinois.edu.*
- *N. Cao is with the Tongji University, Shanghai 200092, China. E-mail: nan.cao@gmail.com.*
- *Y. Xia is with the Facebook AI, Menlo Park, CA 94025 USA. E-mail: yxia@fb.com.*
- *W. Fan is with the Tencent Medical AI Lab, Palo Alto, CA 94301 USA. E-mail: wei.fan@gmail.com.*

Manuscript received 15 Feb. 2019; revised 12 Sept. 2019; accepted 11 Jan. 2020. Date of publication 24 Jan. 2020; date of current version 10 Sept. 2021. (Corresponding author: Meijia Wang.)
Recommended for acceptance by J. Tang.
Digital Object Identifier no. 10.1109/TKDE.2020.2969415

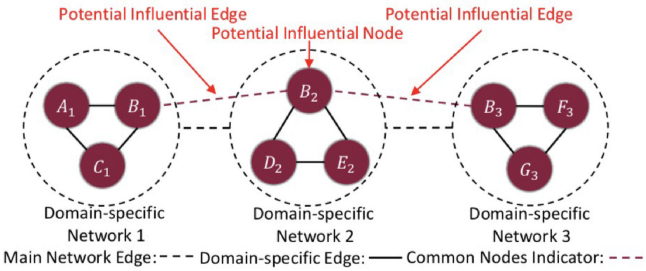


Fig. 1. Example of potential influential edges and node in Network of Networks model.

solving the corresponding optimization problem on large graphs is highly nontrivial.

The main contributions of the paper are summarized as follows.

- *Problem Definition*. We formally define Graph Ranking Auditing Problem and the influence of graph elements as the rate of change in the ranking results upon the perturbation/removal of the graph elements. Then we demonstrate its applicability in multiple ranking algorithms and network structures.
- *Algorithms and Analysis*. We propose a family of fast approximation algorithms to solve the Graph Ranking Auditing Problem, which can achieve a $(1-1/e)$ approximation ratio with a linear complexity.
- *Empirical Evaluations*. We perform extensive experiments on diverse, real-world datasets. The experimental results demonstrate that our proposed methods (a) provide reasonable and intuitive information to help better understand ranking results, and (b) scale linearly w.r.t. the graph size.

The rest of the paper is organized as follows. Section 2 formally defines the auditing problem. Section 3 propose a way to measure the influence of graph elements and presents its application on a set of network structures. Section 4 introduces our proposed algorithms. Then we provide experimental evaluations in Section 5. After reviewing related work in Section 6, we conclude the paper in Section 7.

2 Problem Definition

In this section, we first introduce a table of symbols that will be used throughout the paper (Table 1). Then we formally define the auditing problem.

We use italic uppercase letters G for networks/graphs, calligraphy letters for sets (e.g., $\mathcal{S}$), bold uppercase letters for matrices (e.g., $\mathbf{W}$), bold lowercase letters (e.g., $\mathbf{e}$) for vectors, and lowercase letters for scalars (e.g., c). For matrix indexing convenience, we use the rules similar to Matlab that are shown as follows. We use $\mathbf{W}(i,j)$ to denote the entry of matrix $\mathbf{W}$ at ith row and jth column. We use $\mathbf{W}(i,:)$ and $\mathbf{W}(:,j)$ to denote the ith row and the jth column of matrix $\mathbf{W}$ respectively. For a block matrix $\mathbf{W}$, we use $\mathbf{W}_{(i,j)}$ to denote the (i,j)th block matrix of matrix $\mathbf{W}$. We use prime to denote the transpose of matrix (i.e., $\mathbf{W}'$ is the transpose of matrix $\mathbf{W}$).

The essential idea of unifying the ranking algorithms is to propose a general solution for identifying influential graph elements which serve different objectives in different algorithms. First, we start by introducing the PageRank algorithm and later we will explain how its variants can be unified in the same equation. Given a graph G with n nodes, PageRank essentially solves the following linear system,

$$\mathbf{r} = c\mathbf{W}\mathbf{r} + (1-c)\mathbf{e}, \tag{1}$$

where $\mathbf{e}$ is the teleportation vector with length n and $\mathbf{W}$ is the normalized adjacency matrix of the input graph. In PageRank, $\mathbf{e}$ is chosen as the uniform distribution $\frac{1}{n}\mathbf{1}$; in personalized PageRank, $\mathbf{e}$ is a biased vector which reflects user's preference (i.e., 'personalization') [9]; in random walk with restart [10], all the probabilities are concentrated on a single node. In normalized PageRank, the matrix $\mathbf{W}$ is referred to as the row-normalized adjacency matrix of the graph G. A popular alternative choice is the normalized graph Laplacian matrix.

Here we relaxed the definition of $\mathbf{W}$ to a transition matrix mapped from the graph G by a mapping function θ. In fact, many existing ranking algorithms are equivalent to PageRank and can be formulated as Eq. (1) by defining the corresponding mapping functions. In Section 3, we provide some examples of such ranking algorithms.

In order to guarantee the convergence of Eq. (1) with the transition matrix $\mathbf{W}$, Li *et al*. [11] gives a fixed-point solution $\mathbf{r} = (1-c)(\mathbf{I} - c\mathbf{W})^{-1}\mathbf{e}$ by choosing a factor c such that the largest eigenvalue of $\mathbf{W}$ is less than $1/c$. Consequently, the solution to the linear system problem defined in Equation (1), can be re-written as

$$\mathbf{r} = (1-c)\mathbf{Q}\mathbf{e}, \tag{2}$$

where $\mathbf{Q} = (\mathbf{I} - c\mathbf{W})^{-1}$.

Regarding explainable learning and mining techniques, Pang *et al*. [12] propose a novel notation of influence functions to quantify the impact of each training example on the underlying learning system (e.g., a classifier). The key idea

TABLE 1
Table of Symbols

Symbols	Definitions
$G = \langle \mathbf{A}, \ldots \rangle$	the input network[1]
(i,j)	edge from node i to node j
$\mathbf{A}$	adjacency matrix of the input graph
$\mathbf{W}$	transition matrix of the input graph
$\mathbf{W}(i,j)$	the element at ith row and jth column
$\mathbf{W}(i,:)$	ith row of matrix $\mathbf{W}$
$\mathbf{W}(:,j)$	jth column of matrix $\mathbf{W}$
$\mathbf{W}_{(i,j)}$	the $(i,j)^{\text{th}}$ block in block matrix $\mathbf{W}$
$\mathbf{W}'$	transpose of the matrix $\mathbf{W}$
$\mathbf{W}^{-1}$	inverse of the matrix $\mathbf{W}$
$\mathbf{E}_{ij}$	single-entry matrix with 1 on the $(i,j)^{th}$ element
$\mathbf{e}$	the teleportation vector of PageRank
$\mathbf{r}$	ranking vector of the input network
$\mathbf{r}(i)$	ranking score of the ith node
$\mathrm{Tr}(\mathbf{W})$	Trace of the matrix $\mathbf{W}$
$f(\mathbf{r})$	a loss function over ranking vector $\mathbf{r}$
$\theta(G)$	mapping function from G to a modified matrix
$\mathrm{diag}(\mathbf{r})$	transform vector $\mathbf{r}$ into a diagonal matrix
n	number of elements in the ranking vector
m	number of edges in the input network
c	damping factor in PageRank

1. In this paper, we use 'graph' and 'network' interchangably.

is to trace the model's prediction back to its training data, where the model parameters were derived. In this way, it learns how a perturbation of a single training data will affect the resulting model parameters, and then identifies the training examples that are most responsible for a model's predictions.

Based on the principle outlined in [12], we propose a new method to explain the results of the ranking algorithms. To be specific, we tackle the problem by finding a set of graph elements (e.g., edges, nodes, a subgraph) such that, the ranking result will have the greatest change upon the perturbation/removal of the set of graph elements. Formally, we define Graph Ranking Auditing Problem as follows:

Problem 1 (Graph Ranking Auditing Problem).

Given*: A graph with transition matrix* $\mathbf{W}$*, a teleportation vector* $\mathbf{e}$*, a ranking vector* $\mathbf{r}$*, a loss function* f *over its ranking vector, and an integer budget* k*;*

Find*: a set of* k *influential graph elements (edges, nodes and subgraph) that has the largest impact on the loss function over its ranking vector* $f(\mathbf{r})$*.*

In order to formulate the auditing problem, two key questions need to be answered: (Q1) how to quantitatively measure the influence of an individual graph element w.r.t. the loss function; and (Q2) how to collectively find a set of k graph elements with the maximal influence. We first present our proposed solution for Q1 in Section 3, and then propose three different algorithms for Q2 in Sections 4.1, 4.2 and 4.3, depending on the specific type of graph elements (i.e., edges versus nodes versus subgraphs).

3 Influence Quantification & Examples

In this section, we first formulate the auditing problem and quantify the influence of graph elements w.r.t the loss function. Then we present a general form of the influence function for random walk based ranking algorithms. Finally, we describe how different network structures can be studied in our Graph Ranking Auditing framework.

Problem Formulation. For the ease of description, we first define $\mathbf{r} = \mathrm{pg}(\mathbf{W}, \mathbf{e}, c)$ as the resulting ranking vector given by PageRank and its varients with transition matrix $\mathbf{W}$, teleportation vector $\mathbf{e}$, and damping factor c as the inputs. The intuition behind the proposed methods is to find a set of crucial graph elements (e.g., edges, nodes, subgraphs) whose perturbation/removal from the graph would maximize the change in the loss function and intuitively disturb the ranking result in a desired way. To be specific, let $\mathbf{r} = \mathrm{pg}(\mathbf{W}, \mathbf{e}, c)$ be the ranking vector of the input graph G, and $\mathbf{r}_{\mathcal{S}} = \mathrm{pg}(\mathbf{W}_{\mathcal{S}}, \mathbf{e}, c)$ be the new ranking vector after removing the graph elements in set $\mathcal{S}$ from graph G. We formulate the auditing problem as the following optimization problem,

$$\begin{aligned} \max_{\mathcal{S}} \quad & \left(f(\mathbf{r}) - f(\mathbf{r}_{\mathcal{S}})\right)^2 \\ \text{s.t.} \quad & |\mathcal{S}| = k, \end{aligned} \tag{3}$$

where $f(\mathbf{r})$ is some loss function over the ranking vector $\mathbf{r}$. The choices of possible loss functions are presented in Table 2 and discussed later in this section.

Definition of Influence. To measure how $f(\mathbf{r})$ will change if we perturb/remove a specific graph element, we define its influence as the rate of change in $f(\mathbf{r})$ upon its pertubation/removal.

TABLE 2
Choices of $f(\cdot)$ Functions and Their Derivatives

Descriptions	Functions	Derivatives
L_p norm	$f(\mathbf{r}) = \|\|\mathbf{r}\|\|_p$	$\frac{\partial f}{\partial \mathbf{r}} = \frac{\mathbf{r} \circ \|\mathbf{r}\|^{p-2}}{\|\|\mathbf{r}\|\|_p^{p-1}}$
Soft maximum	$f(\mathbf{r}) = \ln(\sum_{i=1}^{n} \exp(\mathbf{r}_i))$	$\frac{\partial f}{\partial \mathbf{r}} = \left[\frac{\exp(\mathbf{r}_i)}{\sum_{j=1}^{n} \exp(\mathbf{r}_j)}\right]$
Energy norm	$f(\mathbf{r}) = \mathbf{r}'\mathbf{M}\mathbf{r}$	$\frac{\partial f}{\partial \mathbf{r}} = (\mathbf{M} + \mathbf{M}')\mathbf{r}$
Weighted mean	$f(\mathbf{r}) = \mathbf{w}'\mathbf{r}$	$\frac{\partial f}{\partial \mathbf{r}} = \mathbf{w}$

($\mathbf{M}$ in Energy Norm is a Hermitian positive definite matrix.)

Definition 1 (Graph Element Influence). *For a finite simple graph, the influence of an edge* (i, j) *is defined as the derivative of the loss function* $f(\mathbf{r})$ *with respect to the edge, i.e.,* $\mathbb{I}(i, j) = \frac{df(\mathbf{r})}{d\mathbf{W}(i,j)}$.

The influence of a node is defined as the aggregation of all inbound and outbound edges that connect to the node., i.e., $\mathbb{I}(i) = \sum_{j=1, j\neq i}^{n} [\mathbb{I}(i, j) + \mathbb{I}(j, i)] + \sum_{j=1, j=i}^{n} \mathbb{I}(i, j)$. *And the influence of a subgraph is defined as the aggregation of all edges in the subgraph* S, $\mathbb{I}(S) = \sum_{i,j \in S} \mathbb{I}(i, j)$.

We can see that the influence for both nodes and subgraphs can be naturally computed based on the edge influence. Therefore, we will focus on how to measure the edge influence. By the property of the derivative of matrices, we first rewrite the influence $\frac{df(\mathbf{r})}{d\mathbf{W}(i,j)}$ as

$$\frac{df(\mathbf{r})}{d\mathbf{W}} = \begin{cases} \frac{\partial f(\mathbf{r})}{\partial \mathbf{W}} + (\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}})' - \mathrm{diag}(\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}), & \text{if undirected} \\ \frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}, & \text{if directed} \end{cases}. \tag{4}$$

Directly calculating $\frac{df(\mathbf{r})}{d\mathbf{W}(i,j)}$ is hard, and we resort to the chain rule:

$$\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}(i, j)} = \frac{\partial f(\mathbf{r})}{\partial \mathbf{r}'} \frac{\partial \mathbf{r}}{\partial \mathbf{W}(i, j)}. \tag{5}$$

Next, we present the details on how to solve each partial derivative in Eq. (5) one by one.

Computing $\frac{\partial f(\mathbf{r})}{\partial \mathbf{r}'}$. Here we discuss the choices of $f(\cdot)$ function. We list several commonly seen loss functions and their corresponding derivatives in Table 2. Each loss function measures a different aspect of the overall ranking results, and we will conduct experiments on a few of them in Section 5. In the table, L_p *norm* is the most commonly-used vector norm that measures the overall sizes of the vector; *soft maximum* is used to approximate the maximum value of elements in the vector; *energy norm* is a measurement often used in system and control theory to measure the internal energy of vector; *weighted mean* can be tailored to answer specific questions regarding the ranking results.

Computing $\frac{\partial \mathbf{r}}{\partial \mathbf{W}(i,j)}$. Taking the derivative of Eq. (1) with respect to $\mathbf{W}(i, j)$, we obtain

$$\frac{\partial \mathbf{r}}{\partial \mathbf{W}(i, j)} = c\mathbf{W} \frac{\partial \mathbf{r}}{\partial \mathbf{W}(i, j)} + c \frac{\partial \mathbf{W}}{\partial \mathbf{W}(i, j)} \mathbf{r}. \tag{6}$$

The equation is equivalent to Eq. (1) where the teleportation vector is $\frac{c}{1-c}\mathbf{E}_{ij}\mathbf{r}$ since $\frac{\partial \mathbf{W}}{\partial \mathbf{W}(i,j)} = \mathbf{E}_{ij}$ and the gradient vector

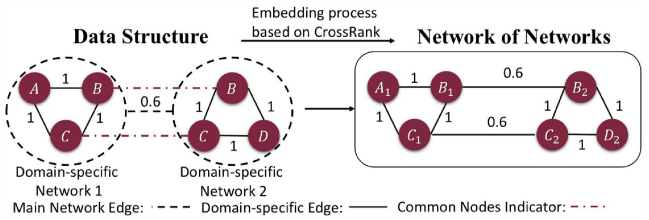


Fig. 2. Example of embedding Network of Networks into plain network by CrossRank.

$\frac{\partial \mathbf{r}}{\partial \mathbf{W}(i,j)}$ is the solution to this linear system, i.e.,

$$\frac{\partial \mathbf{r}}{\partial \mathbf{W}(i,j)} = c\mathbf{Q}\mathbf{e}_i\mathbf{e}'_j\mathbf{r}. \tag{7}$$

Combine everything together, we get the general solution for calculating the influence of an edge (i,j) as follows:

$$\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}(i,j)} = c\left[\frac{\partial f(\mathbf{r})}{\partial \mathbf{r}'}\mathbf{Q}\right](i)\mathbf{r}(j). \tag{8}$$

To obtain the entire gradient matrix $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$, two major computational challenges lie in (1) calculating $\mathbf{Q}$ with $O(n^3)$ time complexity and $O(n^2)$ space complexity to save the matrix of gradients and (2) given $\mathbf{Q}$, calculating $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}(i,j)}$ takes $O(n^2)$ time complexity for one edge and $O(mn^2)$ for all m edges. From Eq. (9), we can re-write it as the following low-rank form

$$\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}} = c\left(\mathbf{Q}'\frac{\partial f(\mathbf{r})}{\partial \mathbf{r}}\right)\mathbf{r}'. \tag{9}$$

The first challenge can be resolved by considering the vector $\mathbf{Q}'\frac{\partial f(\mathbf{r})}{\partial \mathbf{r}}$ as a solution to the linear system in Eq. (1) with personalized teleportation vector $\frac{\partial f(\mathbf{r})}{\partial \mathbf{r}}$. With this in mind, we do not need to calculate $\mathbf{Q}$ explicitly or to save the entire matrix directly. Then the computational complexity for the first matrix product in Eq. (9) is reduced to $O(m)$ time complexity and $O(n)$ space complexity. To extract the element of $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$ at the ith row and the jth column, we simply calculate the product of the ith element in $\mathbf{Q}'\frac{\partial f(\mathbf{r})}{\partial \mathbf{r}}$ and the jth element in $\mathbf{r}$, then scale it by c, which takes $O(1)$ time.

To demonstrate that a variety of network strucutres and ranking algorithms can be studied in our proposed Graph Ranking Auditing framework, here, we explore four prominent examples: (1) plain network; (2) Network of Networks; (3) attributed network and (4) normalized PageRank.

3.1 Example: Unnormalized PageRank

In a plain network, we denote the input graph $G = \langle \mathbf{A} \rangle$ consisting of only the adjacency matrix $\mathbf{A}$ and the largest eigenvalue of $\mathbf{A}$ as $\lambda(\mathbf{A})$. It is straightforward to consider the adjacency matrix $\mathbf{A}$ as the transition matrix $\mathbf{W}$ in Eq. (1) and the mapping function θ is an identity function in this scenario. Here we choose $c = \frac{0.5}{\lambda(\mathbf{A})}$ such that the linear system in Eq. (1) converges to its fixed-point solution.

3.2 Example: Network of Networks

Network of Networks (NoN) data model is first introduced in [13], in which the nodes in the main network (geo-proximity network) can be further represented as domain-specific networks (social networks). In the paper, a ranking algorithm, CrossRank, is designed to rank all the nodes within and across domain-specific networks such that the nodes can be compared in a broader context. The algorithm is further proved to be equivalent to PageRank.

First, we briefly introduce the structure of the model. A Network of Networks is defined as the triplet $G_{non} = \langle \mathcal{A}, \mathbf{M}, \phi \rangle$, where $\mathcal{A} = \{\mathbf{A}_1, \ldots, \mathbf{A}_g\}$ is a set of g adjacency matrices of the corresponding domain-specific networks, $\mathbf{M}$ is a $g \times g$ main network representing the relationship among the domain-specific networks, and ϕ is a one-to-one mapping function for mapping node in $\mathbf{M}$ to domain-specific network. Each node in $\mathbf{M}$, referred to as main nodes, represents a domain-specific network through the mapping function ϕ.

The objective of CrossRank is to find a ranking solution that optimizes the following three problems at the same time: (1) rank smoothly within domain-specific networks; (2) rank accordingly, reflecting the query preference; (3) rank consistently across networks. Jingchao *et al.* [13] derive a fixed-point approach to compute the optimal ranking result according to the above objectives, and they prove that it is equivalent to the linear system in Eq. (1) with a mapping function $\theta_{cr}(\cdot)$ embedding the main network and the domain-specific networks together into a transition matrix. An example of the embedding process on a toy graph is presented in Fig. 2

To align the nodes across networks, CrossRank essentially solves the following linear system

$$\mathbf{r} = \left(\frac{b}{1+2a}\mathbf{A} + \frac{2a}{1+2a}\mathbf{Y}\right)\mathbf{r} + \frac{1-b}{1+2a}\mathbf{e}, \tag{10}$$

where $\mathbf{A}$ is block diagonal matrix with domain specific adjacency matrices $\mathbf{A}_1, \ldots, \mathbf{A}_g$ on its diagonal entries and $\mathbf{Y}$ integrate the information of the main network together with the common nodes across networks. Specifically, $\mathbf{Y}$ is a block matrix whose (i,j)th block $\mathbf{Y}_{(i,j)}$ is an $n_i \times n_j$ binary indicator matrix with $\mathbf{Y}_{(i,j)}(x,y) = \mathbf{M}(i,j)$ if the xth node in $\mathbf{A}_i$ is the yth node in $\mathbf{A}_j$, i.e., common node. The choice of the value of a, b is given by $a = \frac{1}{4\lambda(\mathbf{Y})-2}$ and $b = \frac{1+2a}{2\lambda(\mathbf{A})}$, where $\lambda(\cdot)$ is the leading eigenvalue of the corresponding matrix. To show that CrossRank is equivalent to PageRank, let $\mathbf{W} = \frac{b}{b+2a}\mathbf{A} + \frac{2a}{b+2a}\mathbf{Y}$ and $c = \frac{b+2a}{1+2a}$, then we have Eq (10) identical to Eq (1).

3.3 Example: Attributed Network

Beyond simple graphs and complex network structures, many real-world networks also have rich information embedded in the node and edge attributes. For example, in a social network, nodes may contain information regarding users' demographics and edges may come with interaction information between users. Therefore, it arouses an interesting question that how can we integrate the node and edge attributes information in ranking nodes on large graphs. Further on, we would like to explore what node and edge attributes have the largest impact on the ranking results in an attributed network. In this section, we focus on the attributed network with categorical attributes for both nodes and edges.

Inspired by MAGE [14], in which an edge-augmentation method is proposed to support for categorical node and edge attributes in pattern matching on graphs, we propose a similar attribute-augmentation method to embed all nodes, edges, node attributes and edge attributes in an attributed network into a large plain network.

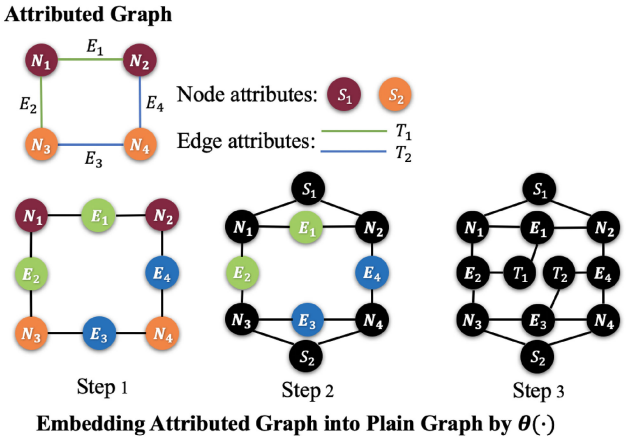


Fig. 3. Example of attribute-augmentation for embedding attributed network into plain graph.

Given an attributed network with n nodes, m edges, n_S categorical node attributes and m_T categorical edge attributes, we denote the attributed network as a triplet $G_{attri} = \langle \mathbf{A}, \mathbf{S}, \mathbf{T} \rangle$, where $\mathbf{A}$ is an $n \times n$ adjacency matrix, $\mathbf{S}$ is an $n \times n_S$ node attribute matrix and $\mathbf{T}$ is an $m \times m_T$ edge attribute matrix. To embed all elements (nodes, edges, node attributes and edge attributes) into a plain network, we first employ the edge-augmentation method based on intuition from the line-graph transformaton [15] to transform the adjacency matrix $\mathbf{A}$ into an $(n+m) \times (n+m)$ adjacency matrix with both nodes and edge-nodes in the matrix, denoted as $\mathbf{E} = \begin{pmatrix} \mathbf{0} & \mathbf{E}^{out} \\ \mathbf{E}^{in} & \mathbf{0} \end{pmatrix}$. Then we take matrix $\mathbf{E}$, $\mathbf{S}$ and $\mathbf{T}$ to form the embedded adjacency matrix in the following form:

$$\mathbf{W} = \begin{bmatrix} \mathbf{0} & \mathbf{E}^{out} & \mathbf{S} & \mathbf{0} \\ \mathbf{E}^{in} & \mathbf{0} & \mathbf{0} & \mathbf{T} \\ \mathbf{S}' & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}' & \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

The embedding process is visualized in Fig. 3 with a toy graph. The embedded matrix is the adjacency matrix of a plain network and can thereby be audited by our proposed method.

3.4 Example: Normalized PageRank

In order to perturb the elements in different networks, we directly make changes on the transition matrix $\mathbf{W}$ by knocking one entry out at a time. However, in some other cases, perturbing an element in a graph may not align with changing an entry in the transition matrix $\mathbf{W}$.

The most typical case is normalized PageRank, where row-normalized adjacency matrix is often used as the transition matrix $\mathbf{W}$ (i.e., $\mathbf{W} = \mathbf{D}_r^{-1}\mathbf{A}$ where $\mathbf{D}_r^{-1}$ is a diagonal matrix with the out-degree/total weights of each row in $\mathbf{A}$ on its diagonal entries). In this case, perturbing an edge (i,j) in the graph not only affects the (i,j)th element in $\mathbf{W}$ but also the entire ith row in it. Therefore, Eqs. (7), (8), and (9) no longer hold for this method. The gradient of the ranking vector over an edge $\mathbf{A}(i,j)$ is given by

$$\frac{\partial \mathbf{r}}{\partial \mathbf{A}(i,j)} = c\mathbf{Q}\mathbf{D}_r^{-1}\mathbf{e}_i(-\mathbf{e}_i'\mathbf{D}_r^{-1}\mathbf{A}\mathbf{r} + \mathbf{e}_j'\mathbf{r}). \tag{11}$$

The gradient of the objective function $f(\mathbf{r})$ with respect to an edge is given by

$$\frac{\partial f(\mathbf{r})}{\partial \mathbf{A}(i,j)} = c\mathbf{u}_{pr}(i) \cdot \left[\mathbf{r}(j) - \left(\mathbf{D}_r^{-1}\mathbf{A}\mathbf{r}\right)(i)\right], \tag{12}$$

where $\mathbf{u}_{pr} = \mathbf{D}_r^{-1}\mathbf{Q}'\frac{\partial f(\mathbf{r})}{\partial \mathbf{r}}$. Accordingly, the matrix of gradients of all edges can be expressed as

$$\frac{\partial f(\mathbf{r})}{\partial \mathbf{A}} = c\left(\mathbf{u}_{pr}\mathbf{r}' + \mathbf{d}_{pr}\mathbf{1}'\right), \tag{13}$$

and we denote $\mathbf{d}_{pr}$ as $\mathbf{d}_{pr} = -\mathrm{diag}(\mathbf{W}\mathbf{r})\mathbf{u}_{pr}$. The vector $\mathbf{u}_{pr}$ can be interpreted as how much the perturbation of an edge would affect the objective function through the source node, $\mathbf{r}$ stands for the impact through the target node and $\mathbf{d}_{pr}$ is the impact through the normalization term.

Another relevant case we would like to cover is the normalized version of Network of Networks in Section 3.2. In [13], the author takes the normalized Laplacian form of $\mathbf{W}$ (i.e., $\tilde{\mathbf{W}} = \mathbf{D}_r^{-\frac{1}{2}}\mathbf{W}\mathbf{D}_r^{-\frac{1}{2}}$). In this case, the matrix of gradients is derived as

$$\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}} = c(\mathbf{u}_{cr}\mathbf{v}_{cr}' + \mathbf{d}_{cr}\mathbf{1}'), \tag{14}$$

where $\mathbf{u}_{cr} = \mathbf{D}_r^{-\frac{1}{2}}\mathbf{Q}'\frac{\partial f(\mathbf{r})}{\partial \mathbf{r}}$, $\mathbf{v}_{cr} = \mathbf{D}_r^{-\frac{1}{2}}\mathbf{r}$ and $\mathbf{d}_{cr} = [\mathrm{diag}(\tilde{\mathbf{W}}\mathbf{r})\mathbf{D}_r^{-\frac{1}{2}} + \mathrm{diag}(\mathbf{D}_r^{-1}\mathbf{r})\tilde{\mathbf{W}}'\mathbf{D}_r^{\frac{1}{2}}]\mathbf{u}_{cr}$. These three vectors can be interpreted as the influence of the edges through the source nodes, target nodes and the normalization term respectively.

It is also worth exploring the influence of the elements in the main network. The gradient matrix on $\mathbf{M}$ is given by

$$\frac{\partial f(\mathbf{r})}{\partial \mathbf{M}} = \frac{2a}{1+2a}\mathbf{U}_{cr}'\mathbf{O}\mathbf{V}_{cr}\emptyset\mathbf{M}, \tag{15}$$

where $\mathbf{U}_{cr} = \mathrm{bdiag}(\mathbf{Q}'\frac{\partial f(\mathbf{r})}{\partial \mathbf{r}})$, $\mathbf{V}_{cr} = \mathrm{bdiag}(\mathbf{r})$, the operation $\mathrm{bdiag}(\cdot)$ diagonalizes the given $n \times 1$ block vector with g blocks into a $n \times g$ block matrix and the operation $\mathbf{U}_{cr}'\mathbf{O}\mathbf{V}_{cr}\emptyset\mathbf{M}$ divide matrix $\mathbf{U}_{cr}'\mathbf{O}\mathbf{V}_{cr}$ element-wise by the non-zero elements in $\mathbf{M}$. With the gradient matrix and the definition of graph element influence, we can thereby audit the network structure on a higher level.

Although we can expect a different point of view in graph ranking auditing by studying these normalized cases, as we can see in the above equations, the influence expressions of the graph elements vary by normalization methods. It is hard to unify the influence function for every normalized version of different ranking algorithms. Moreover, it is noteworthy that with matrix normalization, a negative term is introduced in its derivative. In such case, Theorem 1 (Diminishing Returns Property) in Section 4 no longer holds. Consequently, we can not guarantee a nice approximation ratio to the ground truth with our proposed algorithms in Section 4 for normalized PageRank and its variants.

4 Proposed Algorithms

In this section, we propose a family of algorithms, to solve Graph Ranking Auditing Problem (Problem 1), together with some analysis regarding the effectiveness as well as efficiency of our proposed method.

4.1 Auditing by Edges: AURORA-E

Due to its combinatorial nature, straightforward methods for solving the optimization problem in Eq. (3) are not feasible. The key behind the proposed family of algorithms is based on the diminishing returns property of Problem 1, which is summarized in Theorem 1

Theorem 1 (Diminishing Returns Property of Problem 1). *Given a non-negative gradient matrix, for* any *set of graph elements $\mathcal{S}$, which could be either a set of edges, nodes or subgraphs, in the given graph, its influence measure $\mathbb{I}(\mathcal{S})$ defined in Definition 1 is (a) normalized; (b) monotonically non-decreasing; (c) submodular, where $\mathcal{S} \subseteq \mathcal{E}$.*

Proof. First, we prove the diminishing returns property in the edge case. Let $\mathcal{S}$ be a set of edges and $\mathbb{I}(\mathcal{S}) = \sum_{(i,j)\in\mathcal{S}} \mathbb{I}(i,j)$. It is trivial that if there is no edge selected, the influence is 0. Thus it is normalized. Let $\mathcal{I}, \mathcal{J}, \mathcal{K}$ be three sets of edges and $\mathcal{I} \subseteq \mathcal{J}$. We further define three sets of edges $\mathcal{S}, \mathcal{T}, \mathcal{R}$ as follows: $\mathcal{S} = \mathcal{I} \cup \mathcal{K}$, $\mathcal{T} = \mathcal{J} \cup \mathcal{K}$ and $\mathcal{R} = \mathcal{J} \setminus \mathcal{I}$, then we have

$$\begin{aligned}\mathbb{I}(\mathcal{J}) - \mathbb{I}(\mathcal{I}) &= \sum_{(i,j)\in\mathcal{J}} \mathbb{I}(i,j) - \sum_{(i,j)\in\mathcal{I}} \mathbb{I}(i,j) \\ &= \sum_{(i,j)\in\mathcal{J}\setminus\mathcal{I}} \mathbb{I}(i,j) \\ &= \sum_{(i,j)\in\mathcal{R}} \mathbb{I}(i,j) \geq 0,\end{aligned}$$

which proves that $\mathbb{I}(\mathcal{S})$ is monotonically non-decreasing.

Then, we prove that it is submodular. Define $\mathcal{P} = \mathcal{T} \setminus \mathcal{S}$. We have that $\mathcal{P} = (\mathcal{J} \cup \mathcal{K}) \setminus (\mathcal{I} \cup \mathcal{K}) = \mathcal{R} \setminus (\mathcal{R} \cap \mathcal{K}) \subseteq \mathcal{R} = \mathcal{J} \setminus \mathcal{I}$. Then we have

$$\mathbb{I}(\mathcal{T}) - \mathbb{I}(\mathcal{S}) = \sum_{(i,j)\in\mathcal{P}} \mathbb{I}(i,j) \leq \mathbb{I}(\mathcal{J}) - \mathbb{I}(\mathcal{I}),$$

which proves the submodularity of the edge influence.

Next, we prove the diminishing returns property in the node case. Let $\mathcal{V}$ be a set of nodes and $\mathbb{I}(\mathcal{V}) = \sum_{(i,j)\in\mathcal{S}_\mathcal{V}} \mathbb{I}(i,j)$, where $\mathcal{S}_\mathcal{V}$ is the set of all inbound and outbound edges that connect to the nodes in $\mathcal{V}$. When no node is selected, the sets $\mathcal{V}$ and $\mathcal{S}_\mathcal{V}$ are empty, thus the influence is 0. Let $\mathcal{M}$, $\mathcal{N}$, $\mathcal{O}$ be three sets of nodes and $\mathcal{M} \subseteq \mathcal{N}$. Consequently, $\mathcal{S}_\mathcal{M} \subseteq \mathcal{S}_\mathcal{N}$ and we have

$$\mathbb{I}(\mathcal{N}) - \mathbb{I}(\mathcal{M}) = \mathbb{I}(\mathcal{S}_\mathcal{N}) - \mathbb{I}(\mathcal{S}_\mathcal{M}) \geq 0.$$

To prove it is submodular, let $\mathcal{Q} = (\mathcal{N} \cup \mathcal{O}) \setminus (\mathcal{M} \cup \mathcal{O}) = (\mathcal{N} \setminus \mathcal{M}) \setminus [(\mathcal{N} \setminus \mathcal{M}) \cup \mathcal{O}] \subseteq \mathcal{N} \setminus \mathcal{M}$. Then

$$\mathbb{I}(\mathcal{N} \cup \mathcal{O}) - \mathbb{I}(\mathcal{N} \cup \mathcal{O}) = \mathbb{I}(\mathcal{Q}) \leq \mathbb{I}(\mathcal{N}) - \mathbb{I}(\mathcal{M}).$$

Thus the influence measure is monotonically non-decreasing and submodular for any set of nodes.

Finally, we prove the case for subgraphs. If a set of nodes is an empty set, then the influence of the set is 0 and the influence measure is normalized. Let $\mathcal{D}$, $\mathcal{E}$, $\mathcal{F}$ be three subgraphs and $\mathcal{D} \subseteq \mathcal{E}$. Denote $\mathcal{S}_\mathcal{D}$, $\mathcal{S}_\mathcal{E}$, $\mathcal{S}_\mathcal{F}$ as the sets of edges of the corresponding subgraphs $\mathcal{D}$, $\mathcal{E}$, $\mathcal{F}$ respectively and we have $\mathcal{S}_\mathcal{D} \subseteq \mathcal{S}_\mathcal{E}$. Hence, the influence measure on subgraphs is non-decreasing as

$$\mathbb{I}(\mathcal{E}) - \mathbb{I}(\mathcal{D}) = \mathbb{I}(\mathcal{S}_\mathcal{E}) - \mathbb{I}(\mathcal{S}_\mathcal{D}) \geq 0.$$

Let $\mathcal{H} = (\mathcal{E} \cup \mathcal{F}) \setminus (\mathcal{D} \cup \mathcal{F}) \subseteq \mathcal{E} \setminus \mathcal{D}$. Then

$$\mathbb{I}(\mathcal{E} \cup \mathcal{F}) - \mathbb{I}(\mathcal{D} \cup \mathcal{F}) = \mathbb{I}(\mathcal{H}) \leq \mathbb{I}(\mathcal{E}) - \mathbb{I}(\mathcal{D}).$$

Therefore, we prove that the influence measure on subgraphs is submodular. □

The diminishing returns property naturally leads to a greedy algorithm to obtain a near-optimal solution for solving Problem 1. We first present the algorithm for auditing by edges in this section. The algorithms for auditing by nodes and by subgraphs will be presented in Sections 4.2 and 4.3, respectively.

With the diminishing returns property, we propose AURORA-E (Algorithm 1) algorithm to find top-k influential edges in a graph. The key idea of AURORA-E is to select one edge and update the gradient matrix at each of the k iterations.

Algorithm 1. AURORA-E

Input: The transition matrix $\mathbf{W}$, integer budget k
Output: A set of k edges $\mathcal{S}$ with the highest influence

initialize $\mathcal{S} = \emptyset$;
initialize c (e.g., $c = 1/2\text{maxeigenvalue}(\mathbf{W})$);
calculate ranking $\mathbf{r} = \text{pg}(\mathbf{W}, \mathbf{e}, c)$;
calculate partial gradients $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$ by Eq. (9);
calculate gradients $\frac{df(\mathbf{r})}{d\mathbf{W}}$ by Eq. (4);
while $|\mathcal{S}| \neq k$ **do**
  find $(i,j) = \text{argmax}_{(i,j)}\ \mathbb{I}(i,j)$ with Eq. (4);
  add edge (i,j) to $\mathcal{S}$;
  remove (i,j), and remove (j,i) if undirected;
  re-calculate $\mathbf{r}$, $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$ by Eq. (9), and $\frac{df(\mathbf{r})}{d\mathbf{A}}$ by Eq. (4);
return $\mathcal{S}$;

The effectiveness and efficiency of the proposed AURORA-E are summarized in Lemmas 1 and 2, respectively. We show that AURORA-E finds a $(1 - 1/e)$ near-optimal solution with a linear complexity for ranking algorithms whose gradient matrix on its edges is non-negative as follows.

Lemma 1 (Approximation Ratio of AURORA -E) *Let $\mathcal{S}_k = \{s_1, s_2, \ldots, s_k\}$ represents the set formed by AURORA-E, $\mathcal{O}$ is the optimal solution of Problem 1, $\mathbb{I}(\mathcal{S})$ is the influence defined in Definition 1.*

$$\mathbb{I}(\mathcal{S}_k) \geq (1 - 1/e)\mathbb{I}(\mathcal{O}).$$

Proof. By diminishing returns property, $\forall\ i \leq k$, we have

$$\begin{aligned}\mathbb{I}(\mathcal{O}) \leq \mathbb{I}(\mathcal{O} \cup \mathcal{S}_i) &= \mathbb{I}(\mathcal{S}_i) + \sum_{s\in\mathcal{O}} \Delta(s|\mathcal{S}_i \cup (\mathcal{O} \setminus \{s\})) \\ &\leq \mathbb{I}(\mathcal{S}_i) + \sum_{s\in\mathcal{O}} \Delta(s|\mathcal{S}_i) \leq \mathbb{I}(\mathcal{S}_i) + k\Delta(s_{max}|\mathcal{S}_k),\end{aligned}$$

where $s_{max} = \text{argmax}_{s\in\mathcal{V}\setminus\mathcal{S}_i} \Delta(s|\mathcal{S}_i)$. Then we have

$$\Delta(s_{max}|\mathcal{S}_k) = \mathbb{I}(\mathcal{S}_{i+1}) - \mathbb{I}(\mathcal{S}_i) \geq \frac{1}{k}(\mathbb{I}(\mathcal{O}) - \mathbb{I}(\mathcal{S}_i)).$$

After rearranging the terms, we have

$$\mathbb{I}(\mathcal{O}) - \mathbb{I}(\mathcal{S}_{i+1}) \leq \Big(1 - \frac{1}{k}\Big)(\mathbb{I}(\mathcal{O}) - \mathbb{I}(\mathcal{S}_i))$$
$$\mathbb{I}(\mathcal{O}) - \mathbb{I}(\mathcal{S}_i) \leq \Big(1 - \frac{1}{k}\Big)(\mathbb{I}(\mathcal{O}) - \mathbb{I}(\mathcal{S}_{i-1}))$$
$$\cdots$$
$$\mathbb{I}(\mathcal{O}) - \mathbb{I}(\mathcal{S}_1) \leq \Big(1 - \frac{1}{k}\Big)(\mathbb{I}(\mathcal{O}) - \mathbb{I}(\mathcal{S}_0)).$$

Thus, recursively apply the inequality, we have

$$\mathbb{I}(\mathcal{O}) - \mathbb{I}(\mathcal{S}_k) \leq \Big(1 - \frac{1}{k}\Big)^k (\mathbb{I}(\mathcal{O}) - f(\mathcal{S}_0))$$
$$= \Big(1 - \frac{1}{k}\Big)^k \mathbb{I}(\mathcal{O}) \leq \frac{1}{e}\mathbb{I}(\mathcal{O}).$$

Thus we have $(1 - 1/e)\mathbb{I}(\mathcal{O}) \leq \mathbb{I}(\mathcal{S}_{i+1})$. □

It is noteworthy that the guarantee of the efficiency no longer holds for gradient matrix with both positive and negative values. For example, when we take the row normalization or Laplacian normalization of the adjacency matrix in the mapping function as mentioned in Section 3.4, a negative term is introduced as the influence through the normalization term, and the gradient matrix is therefore no longer non-negative.

Algorithm 2. AURORA-N

Input: The transition matrix $\mathbf{W}$, integer budget k
Output: A set of k nodes $\mathcal{S}$ with highest influence

initialize $\mathcal{S} = \emptyset$;
initialize c (e.g., $c = 1/2\text{maxeigenvalue}(\mathbf{W})$);
calculate ranking $\mathbf{r} = \text{pg}(\mathbf{W}, \mathbf{e}, c)$;
calculate partial gradients $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$ by Eq. (9);
calculate gradients $\frac{df(\mathbf{r})}{d\mathbf{W}}$ by Eq. (4);
while $|\mathcal{S}| \neq k$ **do**
  find $v_i = \text{argmax}_i\ \mathbb{I}(i)$;
  add v_i to $\mathcal{S}$;
  remove all inbound and outbound edges of v_i;
  re-calculate $\mathbf{r}$, $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$ by Eq. (9), and $\frac{df(\mathbf{r})}{d\mathbf{W}}$ by Eq. (4);
return $\mathcal{S}$;

Lemma 2 (Time and Space Complexities of AURORA-E) *Algorithm 1 is $O(mk)$ in time and $O(m+n)$ in space, where m and n are the numbers of edges and nodes in the input graph; and k is the budget.*

Proof. It takes $O(m)$ time complexity to calculate $\mathbf{r}$ and $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$ by applying power iterations. In the while-loop, we find the edge with the greatest influence by traversing all edges, which takes $O(m)$ time. Time spent to re-calculate $\mathbf{r}$ and $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$ remains the same as $O(m)$. Since the body inside the loop will run k times, the overall time complexity is $O(mk)$. In Algorithm 1, it takes $O(m)$ space to save the sparse adjacency matrix $\mathbf{W}$ and $O(n)$ space to save the ranking vector $\mathbf{r}$ and column vector $\mathbf{Q}'\mathbf{r}$ in Eq. (9). Therefore it has $O(m+n)$ space complexity. □

4.2 Auditing by Nodes: AURORA-N

By Theorem 1, the influence of nodes also enjoys the diminishing returns property for non-negative gradient matrix. Following this, we propose a greedy algorithm AURORA-N (Algorithm 2) to find a set of top-k influential nodes in a graph with $(1 - 1/e)$ approximation ratio with linear complexity. The efficiency of the proposed AURORA-N is summarized in Lemma 3.

Lemma 3 (Time and Space Complexities of AURORA -N). *Algorithm 2 is $O(mk)$ in time, and $O(m+n)$ in space, where m and n are the numbers of edges and nodes in the input graph; and k is the budget.*

Proof. It takes $O(m)$ time complexity to calculate $\mathbf{r}$ and $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$ by applying power iterations. In the while-loop, we calculate the influence of nodes and find the node with the greatest influence by traversing all edges, which takes $O(m)$ time. Time spent to re-calculate $\mathbf{r}$ and $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$ remains the same as $O(m)$. Since the body inside loop will run k times, the overall time complexity is $O(mk)$. In Algorithm 2, it takes $O(m)$ space to save the sparse adjacency matrix $\mathbf{W}$ and $O(n)$ space to save the ranking vector $\mathbf{r}$ and column vector $\mathbf{Q}'\mathbf{r}$ in Eq. (9). Therefore it has $O(m+n)$ space complexity. □

4.3 Auditing by Subgraphs: AURORA-S

Here, we discuss how to select an influential subgraph with k nodes and we focus on the vertex-induced subgraph. With the diminishing returns property (Theorem 1) in mind, we propose AURORA-S (Algorithm 3) to greedily identify the influential subgraph with $(1 - 1/e)$ approximation ratio with a linear complexity. The efficiency of the proposed AURORA-S is summarized in Lemma 4.

Algorithm 3. AURORA-S

Input: The transition matrix $\mathbf{W}$, output size k
Output: A vertex-induced subgraph of k nodes $\mathcal{S}$ with highest influence

initialize $\mathcal{S} = \emptyset$;
initialize c (e.g., $c = 1/2\text{maxeigenvalue}(\mathbf{W})$);
calculate ranking $\mathbf{r} = \text{pg}(\mathbf{W}, \mathbf{e}, c)$;
calculate partial gradients $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$ by Eq. (9);
calculate gradients $\frac{df(\mathbf{r})}{d\mathbf{A}}$ by Eq. (4);
while $|\mathcal{S}| \neq k$ **do**
  find $(i,j) = \text{argmax}_{(i,j)}\ \mathbb{I}(i,j)$;
  if $|\mathcal{S}| + 2 \leq k$ **then**
    add v_i and v_j to $\mathcal{S}$;
  else
    find the endpoint v with higher gradient;
    if $v \notin \mathcal{S}$ **then**
      add v to $\mathcal{S}$;
    else
      add the other endpoint to $\mathcal{S}$;
  remove all edges in $\mathcal{S}$;
  re-calculate $\mathbf{r}$, $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$ by Eq. (9), and $\frac{df(\mathbf{r})}{d\mathbf{W}}$ by Eq. (4);
return $\mathcal{S}$;

Lemma 4 (Time and Space Complexities of AURORA -S) *Algorithm 3 is $O(mk)$ in time and $O(m+n)$ in space, where m and n are the numbers of edges and nodes in the input graph; and k is the budget.*

TABLE 3
Statistics of the Datasets

Category	Network	Type	Nodes	Edges
Social	Karate	U	34	78
	Dolphins	U	62	159
	WikiVote	D	7,115	103,689
	Pokec	D	1,632,803	30,622,564
Collaboration	GrQc	U	5,242	14,496
	DBLP	U	42,252	420,640
	NBA	U	3,923	71,581
	cit-DBLP	D	12,591	49,743
	cit-HepTh	D	27,770	352,807
	cit-HepPh	D	34,546	421,578
Physical	Airport	D	1,128	18,736
NoN	DBLP-NoN	U	259,822	622,532
	PPI-NoN	U	798,185	4,553,022
Attributed	DBLP-attr	D	1,065,882	3,158,894
	CORA-attr	D	9,570	60,074
Others	Lesmis	U	77	254
	Amazon	D	262,111	1,234,877

(In Type, U means undirected graph; D means directed graph.)

Proof. It takes $O(m)$ time complexity to calculate $\mathbf{r}$ and $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$ by applying power iterations. In the while-loop, we find the edge with the greatest influence by traversing all edges, which takes $O(m)$ time. Time spent to re-calculate $\mathbf{r}$ and $\frac{\partial f(\mathbf{r})}{\partial \mathbf{W}}$ remains the same as $O(m)$. Since the body inside loop will run k times, the overall time complexity is $O(mk)$. In Algorithm 3, it takes $O(m)$ space to save the sparse adjacency matrix $\mathbf{W}$ and $O(n)$ space to save the ranking vector $\mathbf{r}$ and column vector $\mathbf{Q}'\mathbf{r}$ in Eq. (9). Therefore it has $O(m+n)$ space complexity. □

5 Experimental Evaluation

In this section, we evaluate the proposed Aurora algorithms. All experiments are designed to answer the following two questions:

- *Effectiveness.* How effective are the proposed Aurora algorithms in identifying key graph elements w.r.t. the ranking results?
- *Efficiency.* How efficient and scalable are the proposed Aurora algorithms?

5.1 Setup

Datasets. We test our algorithms on a diverse set of real-world network datasets. All datasets are publicly available. The statistics of these datasets are listed in Table 3.

- Social Networks. Here, nodes are users, and edges indicate social relationships. Among them, *Karate* [16] is a well-known network dataset of a university karate club collected by Wayne Zachary in 1977. *Dolphins* [17] is an undirected social network of frequent associations between dolphins in a community living off Doubtful Sound, New Zealand. *WikiVote* [18] is generated by Wikipedia voting data from the inception of Wikipedia till January 2008. *Pokec* [19] is a popular online social network in Slovakia.
- Collaboration Networks. Here, nodes are individuals and two people are connected if they have collaborated. We use the collaboration network in the field of General Relativity and Quantum Cosmology (*GrQc*) in Physics from arXiv preprint archive.[2] *DBLP*[3] is a co-authorship network from DBLP computer science bibliography. And *NBA* [20] is a collaboration network of NBA players from 1946 to 2009. *cit-DBLP* [21] is the citation network of DBLP, a database of scientific publications such as papers and books. Each node in the network is a publication, and each edge represents a citation of a publication by another publication. *cit-HepTh* [22] is an ArXiv HEP-TH (High Energy Physics - Theory) citation network. The data covers papers from January 1993 to April 2003. If a paper i cites paper j, there is a directed edge from i to j. *cit-HepPh* [22] is an ArXiv HEP-PH (High Energy Physics - Phenomenology) citation network. The data covers papers from January 1993 to April 2003. If a paper i cites paper j, there is a directed edge from i to j.
- Physical Infrastructure Networks. This category refers to the networks of physical infrastructure entities. Nodes in them correspond to physical infrastructure, and edges are connections. *Airport*[4] is a dataset of airline traffic. Each node represents an airport in the United States, an edge (i, j) represents the airline from i to j while the edge weight stands for the normalized number of passengers.
- Network of Networks. Here, each node in the main network can be further represented as a domain-specific network. In *DBLP-NoN* dataset [23], the main network consists of 121 conferences and the edges represents the similarity across conferences which are generated by [24]. The domain-specific networks are the co-author networks with a total of 259,822 nodes. The within-layer edges are weighted by the number of collaborative works between authors in the area, and the cross-layer edges represent the same author with publications in different areas. The tissue-specific protein interaction NoN (*PPI-NoN*) is first introduced in [13], which consists of a disease similarity network (the main network) with 170 diseases. In the main network, each node is a tissue-specific molecular network corresponding to the disease with a total of 9,998 proteins.
- Attributed Graphs This category refers to attributed graphs that are embedded in plain graphs. The *DBLP-attr*, with a total of 1,065,882 embedded nodes, consists of 162,932 author-nodes, 902,708 edge-nodes, 121 node-attribute-nodes and 121 edge-attribute-nodes. The *CORA-attr* is based on the well-known CORA dataset [25], with a total of 9,570 nodes. It consists of 2,708 nodes representing scientific publications, 5,278 edge-nodes embedded from the citation network and 1,433 node-attribute-nodes transformed

2. https://arxiv.org/
3. http://dblp.uni-trier.de/
4. https://www.transtats.bts.gov/

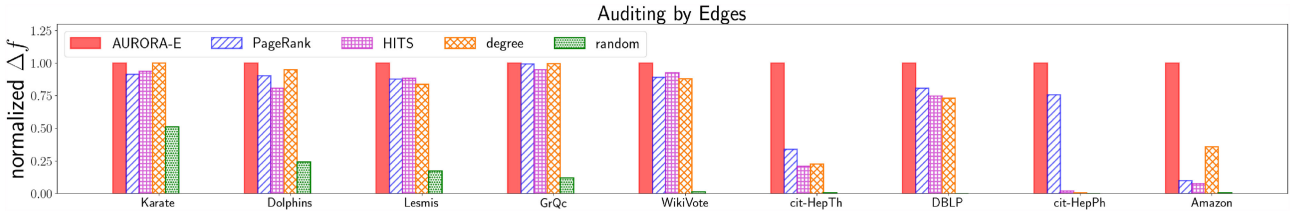


Fig. 4. Auditing results by edges on plain graphs with unnormalized PageRank. Budget $k = 10$. Higher is better. Best viewed in color.

from a 0/1-valued word vector describing the corresponding publication.

- OTHERS. This category contains networks that do not fit into the above categories. *Lesmis* [26] is a network of co-appearances of characters in Victor Hugo's novel "*Les Miserables*". A node represents a character and an edge connects a pair of characters if they both appear in the same chapter of the book. *Amazon* [27] is a co-purchasing network collected by crawling Amazon website. It is based on the *Customers Who Bought This Item Also Bought* feature.

Baseline Methods. We compare our proposed methods with several baseline methods, which are summarized as follows.

- *Random Selection (random)*. Randomly select k elements and calculate the change by removing them.
- *Top-k Degrees (degree)*. We first define the degree of an edge (u, v) as follows,

$$d(u,v) = \begin{cases} (d(u) \times d(v)) \times \max\limits_{i \in \{u,v\}} d(i), & \text{if undirected} \\ (d(u) \times d(v)) \times d(u), & \text{if directed,} \end{cases}$$

where $d(u)$ represents the degree of node u.

To audit by graph elements, we select k elements with the highest degrees. For edges, we select k edges with the highest edge degrees defined above; for nodes, we select k nodes with the highest node degrees; for subgraphs, we form a vertex-induced subgraph from k nodes with the highest degrees.

- *PageRank*. We first define the PageRank score of an edge (u, v) as follows,

$$pg(u,v) = \begin{cases} (\mathbf{r}(u) \times \mathbf{r}(v)) \times \max\limits_{i \in \{u,v\}} \mathbf{r}(i), & \text{if undirected} \\ (\mathbf{r}(u) \times \mathbf{r}(v)) \times \mathbf{r}(u), & \text{if directed} \end{cases},$$

where $\mathbf{r}(u)$ is the PageRank score of node u.

To audit by graph elements, we select k elements with the highest PageRank scores. That is, for edges, we select k edges with the highest PageRank scores defined above; for nodes, we select k nodes with highest PageRank scores; for subgraphs, we form a vertex-induced subgraph from k nodes with the highest PageRank scores.

- *HITS*. We first define HITS score of an edge (u, v) and node u as follows,

$$HITS(u,v) = hub(u) \times hub(v) + auth(u) \times auth(v)$$
$$HITS(u) = hub(u) + auth(u),$$

where $hub(u)$ and $auth(u)$ represent the hub score and authority score of node u, respectively.

To audit by graph elements, we select k elements with the highest HITS scores. That is, for edges, we select k edges with the highest HITS scores defined above; for nodes, we select k nodes with the highest HITS scores; for subgraphs, we form a vertex-induced subgraph from k nodes with the highest HITS scores.

Metrics. Here, we choose the loss function to be squared L_2 norm. We quantify the performance of auditing by the goodness score Δf of the graph elements S found by the corresponding algorithms. The goodness score we measure is defined as

$$\Delta f = \left| f(\mathbf{r}/\sum_{i=1}^{n} \mathbf{r}) - f(\mathbf{r}_S/\sum_{i=1}^{n} \mathbf{r}_S) \right|. \tag{16}$$

Repeatability and Machine Configuration. All datasets are publicly available. We will release the code of our proposed algorithms upon the publication of the paper. All experiments are performed on a Windows 10 machine with 6 Intel i7-8700 CPU cores at 3.2 GHz and 32 GB RAM. All codes are written in Python 3.6.

5.2 Quantitative Comparison

We perform effectiveness experiments on our proposed algorithms and compare them with the baseline methods. We set k from 1 to 10 and find k influential edges and nodes, respectively. For subgraph, e set k only from 2 to 10 to find an influential subgraph of size-k. This is because a vertex-induced subgraph with only 1 node does not contain any edge and therefore is meaningless for graph ranking auditing. It is worth pointing out that searching a ground-truth with k most influential elements is prohibitively expensive due to its combinatorial nature. For example, even if we use the small *Lesmis* dataset, it will take over a day to find ground-truth with $k = 5$. Therefore, we do not include ground-truth with k influential elements on all datasets.

Auditing on Plain Network. The results of quantitative comparison on plain network across 9 different datasets are shown from Figs. 4, 5, and 6, and from Figs. 9 and 10. From those figures, we have the following observation that our family of AURORA algorithms consistently outperform other baseline methods on all datasets.

Auditing on Network of Networks. The results of quantitative comparison on Network of Networks with two datasets are shown in Fig. 7. From the figure, we observe that AURORA-E performs better than other baseline methods, but AURORA-N and AURORA-S are slightly outperformed by PageRank and HITS on *DBLP-non* dataset respectively. By further investigating in why AURORA-S is outperformed, we notice that the HITS

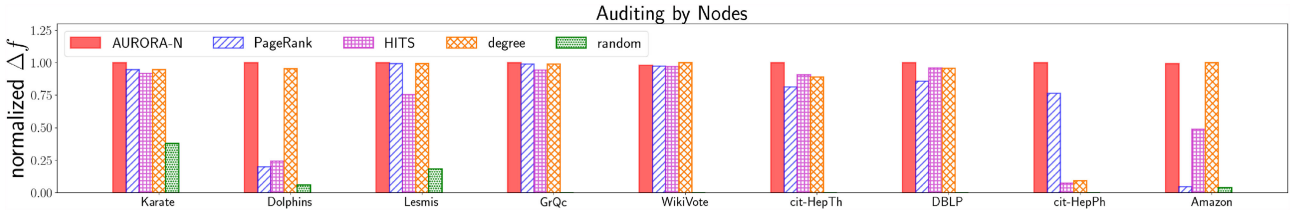


Fig. 5. Auditing results by nodes on plain graphs with unnormalized PageRank. Budget $k = 10$. Higher is better. Best viewed in color.

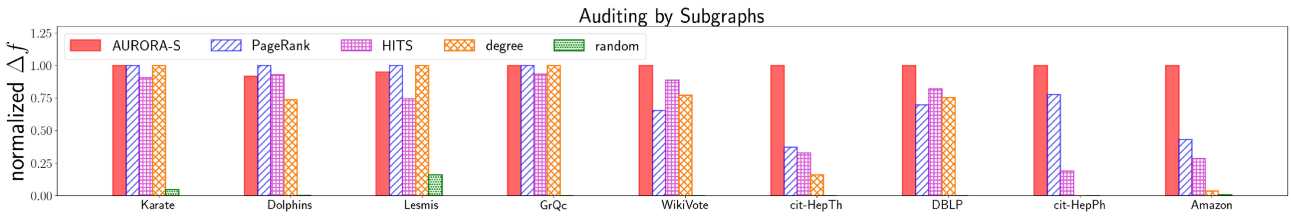


Fig. 6. Auditing results by subgraphs on plain graphs with unnormalized PageRank. Budget $k = 10$. Higher is better. Best viewed in color.

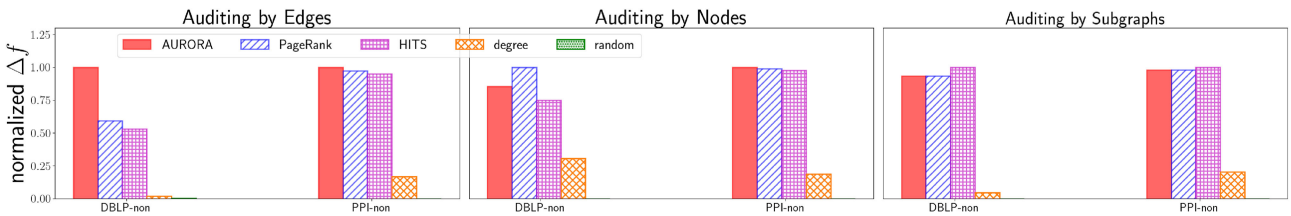


Fig. 7. Auditing results on Network of Networks. Budget $k = 10$. Higher is better. Best viewed in color.

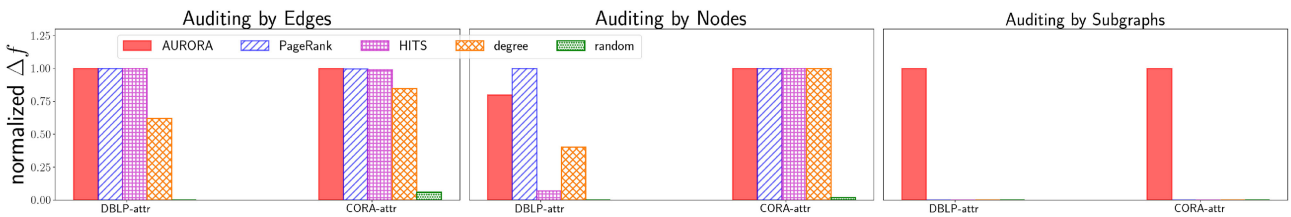
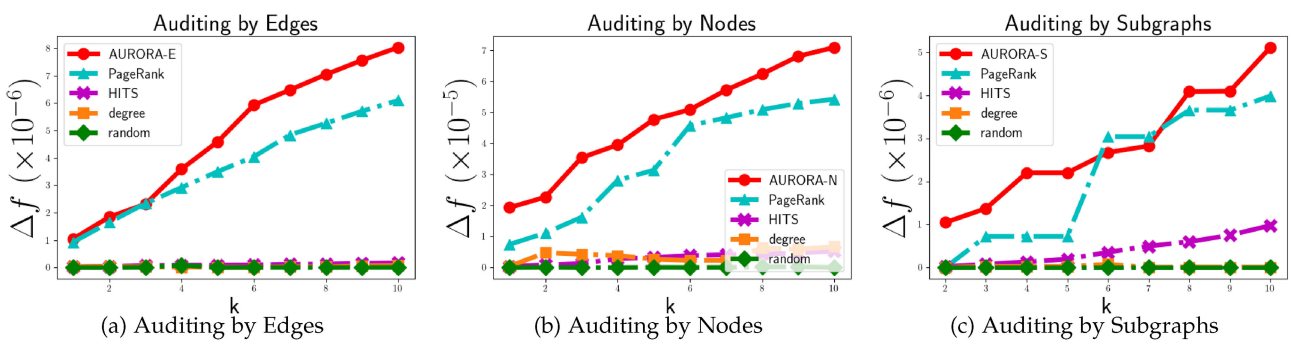


Fig. 8. Auditing results on attributed graphs. Budget $k = 10$. Higher is better. Best viewed in color.

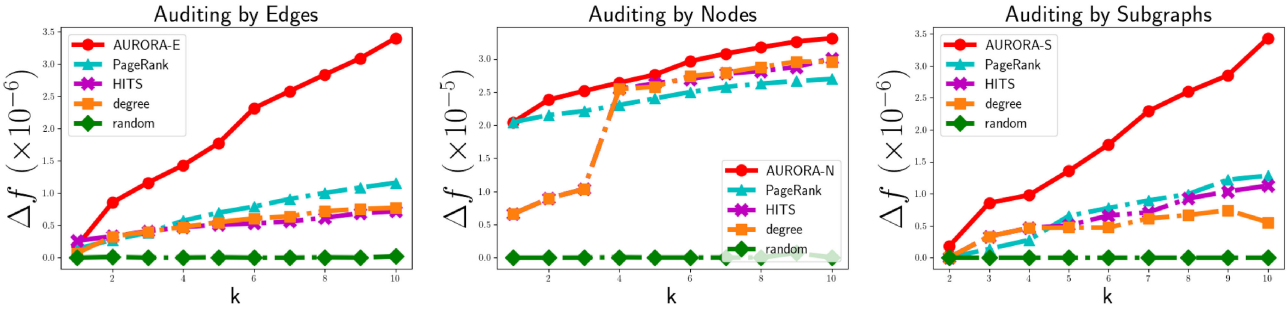


(a) Auditing by Edges (b) Auditing by Nodes (c) Auditing by Subgraphs

Fig. 9. Effect of k on auditing results (cit-HepPh Dataset). Higher is better. Best viewed in color.

method selects all ten nodes from the same domain-specific network to form the subgraph while AURORA-S selects nodes from across several domain-specific networks. Recall that in a Network of Networks model, most of the connections/edges lie within domain-specific networks and cross-network connections only occur when the same entity appears in multiple domain-specific networks. Therefore, the HITS method, by measuring the authority and hub scores of nodes, happens to find a set of nodes that are firmly connected. Here AURORA-S selects a set of nodes that are influential if we remove all the edges they have but do not work so well regarding the subgraph they consisted of.

Auditing on Attributed Graphs. The results of quantitative comparison on attributed graphs are shown in Fig. 8. We observe that our algorithms produce the same results as some of the baseline methods in finding influential edges and nodes. We conclude that it is due to the extremely unbalanced degree distribution on the embedded graphs. In *DBLP-attr*



(a) Auditing by Edges (b) Auditing by Nodes (c) Auditing by Subgraphs

Fig. 10. Effect of k on auditing results (cit-HepTh Dataset). Higher is better. Best viewed in color.

TABLE 4
Case Study Result on DBLP Datasets

DBLP		DBLP-NoN		DBLP-attr	
PageRank	AURORA-N	PageRank	AURORA-N	PageRank*	AURORA-N*
Rakesh Agrawal	Rakesh Agrawal	VLDB: Spiros Papadimitriou	VLDB: Spiros Papadimitriou	KDD	**VLDB**
Michael J. Carey	Michael J. Carey	VLDB: H. V. Jagadish	VLDB: H. V. Jagadish	VLDB	KDD
H. V. Jagadish	H. V. Jagadish	VLDB: Timos K. Sellis	VLDB: Timos K. Sellis	ICDE	ICDE
Joseph M. Hellerstein	Joseph M. Hellerstein	KDD: Jure Leskovec	**VLDB: Flip Korn**	SIGMOD	SIGMOD
Yannis E. Ioannidis	Gerhard Weikum	ICDE: Spiros Papadimitriou	**VLDB: Agma J. M. Traina**	SDM	SDM
Gerhard Weikum	**Jure Leskovec**	VLDB: Flip Korn	**VLDB: Jimeng Sun**	ICDM	ICDM

(In DBLP-attr, *all the results are the node-attributes in the network)*

dataset, 42 percent of the edges in the embedded graph link to the 121 node-attribute-nodes and 121 edge-attribute-nodes. In *CORA-attr* dataset, 82 percent of the edges link to the 1433 node-attribute-nodes. We believe that the PageRank method having a better result than AURORA-S on the DBLP-attr dataset is also due to this reason. Though our proposed embedding method causes the extreme degree distribution in these cases, we find our algorithms work well in finding influential subgraphs and produce meaningful results as discussed in our case study on *DBLP-attr* dataset in Section 5.3.1. We believe it is mainly because the high degrees come with these attributes nodes correctly reflect the popularity of the features among the original nodes and edges. Since ranking the nodes, edges from the original graph and the attribute-nodes all together may seem unfair at some point, we can simply discriminate them by finding the influential elements among each node type respectively.

By comparing the quantitative performance of our AURORA algorithms across the three kinds of network structures, we have the conclusion that AURORA algorithms consistently outperform our baseline methods on plain network but are unstable on the NoN datasets and the attributed graph datasets. We believe that the uncertainty in performance is due to the different graph distributions that are more likely to occur in the Network of Networks and attributed graphs. In Network of Networks model, the connectivity among nodes are dense within domain-specific networks and are extremely sparse across domain networks. As for attributed graphs, the embedded attribute-nodes tend to have much higher degrees than other nodes. Therefore, in order to improve the quantitative performance of AURORA algorithms, further work is needed to study the relation between the performance of our algorithms and the graph structures.

5.3 Qualitative Comparison

In order to show that our proposed AURORA methods can provide intuitive and reasonable explanations, we conduct case studies on three real-world datasets, consisting of two collaboration networks (i.e., *DBLP* dataset and *NBA* dataset) and a physical infrastructure network (i.e., *Airport* dataset).

5.3.1 *Case Studies on* DBLP *Datasets*

A nature use case of AURORA algorithms is sense-making in graph proximity. We construct three different types of network structure from DBLP computer science bibliography to test our algorithms. For a plain network, we directly construct the graph based on co-authorship among authors with unweighted edges. For Network of Networks, we use the 121 conferences in DBLP as the main networks and the number of collaborated works as weighted edges between authors in each domain-specific networks. We then construct the attributed graph with the number of publications in each conference as the node attributes of authors and the number of collaborated works as the edge attributes among users.

We perform AURORA-N and PageRank with $k = 6$ on all three DBLP datasets. Here we use a personalized teleportation vector with the query node `Christos Faloutsos`. In this case, the top-ranked scholars in the resulting ranking vector $\mathbf{r}$ form the proximity (i.e., 'neighborhood') of the query node (i.e., who are most relevant to `Christos Faloutsos`). Consequently, the nodes selected by an auditing algorithm are influential in making/maintaining the neighborhood of the query node. The results are summarized in Table 4.

Comparing the results of AURORA-N with PageRank on the plain network, 5 of them are the same while AURORA-N selects `Jure Leskovec` instead of `Yannis Ioannidis`.

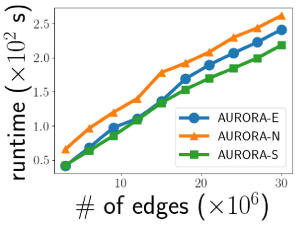


Fig. 11. running time versus the number of edges on Pokec dataset.

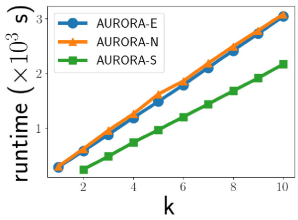


Fig. 12. running time versus number of k on Pokec dataset.

This result is consistent with the intuition, since Jure Leskovec, as a former student of Christos Faloutsos with lots of joint publications, plays a more prominent role in the neighborhood of Christos Faloutsos by sharing more common collaborators.

As for the query results on *DBLP-non*, two methods give the same set of collaborators in the first three places and differ in the last threes. AURORA-N recommends all six collaborators in VLDB network, which is persuasive as the query node Christos Faloutsos built a strong connection network and published a great amount of papers in VLDB in his early career. Our method ranks Flip Korn, Agma J.M. Traina and Jimeng Sun in the fourth to sixth places based on their many collaborations with the query node in VLDB network as well as in many other conferences and journals. The results of the baseline method also make sense considering the number of collaborations bettween the recommended nodes and the query node.

In the attributed graph, AURORA-N and PageRank give completely the same set of nodes on *DBLP-attr* but slightly differ in order. AURORA-N put VLDB in the first place and KDD in the second place in spite that the query node Christos Faloutsos has more publications in the latter. It is consistent with the results on *DBLP-non* and can be explained similarly. Our method recognizes VLDB as the most important attribute to the ranking result of the query node. Given that the auditing results are all embedded node-attribute-nodes, we conclude that this is due to the particular network distribution of the embedded attributed graph. In fact, we can also look for the most influential author nodes and collaboration edges in the attributed network by selecting the influential graph elements from the nodes or edge-nodes subset in the augmented graph.

5.3.2 *Case Studies on* Airport *Dataset*

Another compelling use case of our AURORA algorithms is to find influential edges and nodes in a given graph. To demonstrate that our algorithms are indeed able to provide intuitive information, we test our algorithms on the *Airport* dataset. This dataset was manually created from commercial airline traffic data in 2017, which is provided by the United States Department of Transportation. More detailed description and statistics of this dataset can be found in Section 5.1. We perform AURORA-E and AURORA-N to reveal the most influential airlines (edges in the graph) and airports (nodes in the graph) across the United States with $k = 7$.

Edges selected by AURORA-E are ATL-LAX, LAX-ATL, ATL-ORD, ORD-ATL, ATL-DEN, DEN-ATL and LAX-ORD. In contrast, PageRank selects *ATL-LAS* instead of DEN-ATL and *ATL-DFW* instead of LAX-ORD. DEN-ATL plays a more important role in determining the centrality (e.g., PageRank) of other airports. This is because DEN serves as one of the busiest hub airports that connects West coast and East coast; while *ATL-LAS* is less important in that regard, considering the existence of ATL-LAX and ATL-PHX. Comparing LAX-ORD and *ATL-DFW*, LAX-ORD directly connects Los Angeles and Chicago, both of them are largest cities in the United States.

In the scenario of node-auditing, AURORA-N selects ATL, LAX, ORD, DFW, DEN, LAS and CLT. In contrast, PageRank selects *SFO* instead of CLT. CLT seems to be a more reasonable choice because it serves as a major hub airport, the 6th busiest airport by FAA statistics, to connect many regional airports around States like North Carolina, South Carolina, Virginia, West Virginia, etc. Compared with CLT, *SFO* is less influential in that regard, mainly due to the following two reasons: (1) it ranks after CLT (7th versus 6th) in the list of busiest airports by FAA statistics; (2) due to the location proximity of *SFO* to LAX and SJC, even if this node is perturbed (i.e., absent), many surrounding airports (especially regional airports in California) could still be connected via LAX and SJC.

5.3.3 *Case Studies on* NBA *Dataset*

In a collaboration network, a subgraph can be naturally viewed as a team (e.g., sports team). From this perspective, AURORA-S has the potential to find teammates of a player. We set the query node as Tracy McGrady. Since there are average 14 players for each team in NBA, we set the budget $k = 14$. Comparing the results by AURORA-N and PageRank, 13 of them are the same. However, AURORA-N selects Rafer Alston instead of Steven Hunter. As Tracy's teammate, we believe Rafer Alston is a more important collaborator and teammate mainly because of the following two reasons: (1) Rafer Alston played more seasons with Tracy McGrady than Steven Hunter (4 seasons, 191 games versus 3 seasons, 129 games); and (2) he played more games in the starting lineups with Tracy McGrady than Steven Hunter (187 games versus 47 games).

5.4 Efficiency Results

We show the running time versus number of edges m and budget size k on *Pokec* dataset in Figs. 11 and 12. We can see that the proposed AURORA algorithms scale linearly w.r.t. m and k, respectively. This is consistent with our complexity analysis that the family of AURORA algorithms are linear with respect to the number of edges and the budget.

5.5 Visualization

To better understand the auditing results, we developed a prototype system with D3.js to represent the influence of graph elements visually. In the system, we use the strength

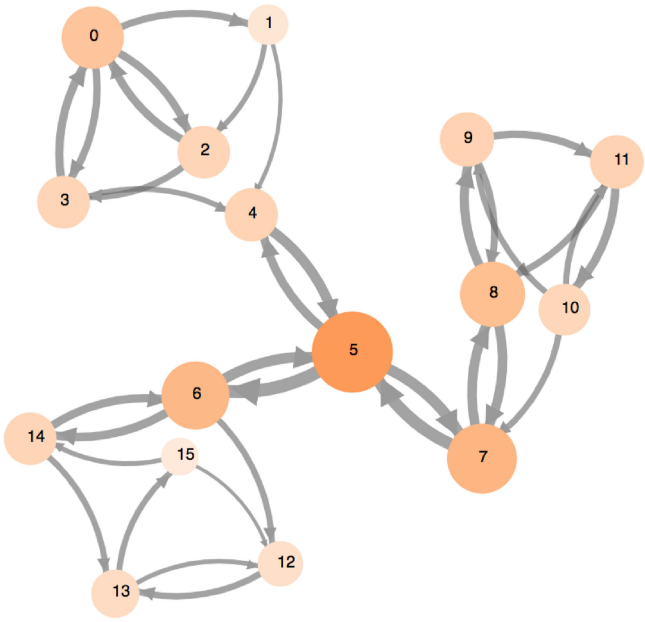


Fig. 13. Visualization of toy graph on the visualization system. Best viewed in color.

of line to represent the gradient of an edge and use the size and color for the gradient of nodes. An example of visualizing hand-crafted toy graph is shown in Fig. 13. As shown in the figure, Node 5 is the most influential node selected by AURORA-N, and edges around Node 5 are more influential than other edges according to AURORA-E, both of which are consistent with our intuition.

6 Related Work

In this section, we briefly review the related work, from the following two perspectives, including (1) graph ranking and (2) applications of graph ranking auditing.

Graph Ranking. Regarding graph ranking, PageRank [1] and HITS [28] are probably the most well-known and widely used algorithms. PageRank measures the importance of nodes as a stationary distribution of random walks. HITS assumes that each node has two scores: *hub* and *authority*. A node has a high hub score if it links to many nodes with high authority scores, and a node has a high authority score if it links to many nodes with high hub scores. Many variants of PageRank and HITS have been developed in the literature. To name a few, in [29], the authors study the stability of PageRank and HITS, based on which they propose two new algorithms (Randomized HITS and Subspace HITS). Ding *et al.* [30] provide a unified ranking method for HITS and PageRank. In [9], Haveliwala *et al.* propose the well known *personalized PageRank* by replacing the uniform teleportation vector with a biased personalized topic-specific vector; while random walk with restart [10] concentrates all teleportation probabilities to a single node. Other random walk based graph ranking methods include [31] and many more.

Applications of Graph Auditing. The idea of graph auditing is first introduced in the field of sensitivity analysis on graph data in order to measure the robustness of graph statistics with respect to noise and perturbation. [32], [33], [34] and [35] use various vertex removal strategies to study the behavior of social networks, web graphs and generated graph data with respect to centrality measures, stochastic quantifiers, and shortest path distribution.

How to enhance interpretability and robustness of graph mining models attracts many research interests in recent years. Dai *et al.* [36] propose reinforcement learning based attack methods targeting a family of Graph Neural Network models for node representation learning. Wang *et al.* [37] propose a greedy attack approach to Graph Convolutional Network models. Zügner *et al.* [38] study both the structure attacks and the feature attacks for several node classification models targeting node classification on attributed graphs under the framework of unnoticeable perturbation. Poisoning attack on DeepWalk and LINE models for node embedding is first investigated by Sun *et al.* [39]. Chen [40] and Zhou [41] develop two novel methods for attacking link prediction on graphs.

In terms of the interpretability of graph ranking, Scott *et al.* [42] present a web-based prototype for an explainable ranking algorithm in multi-layered networks. Varadarajan *et al.* [43] propose a way to explain the ranking results of ObjectRank by computing a subgraph that reflects the authority flows in the graph regarding the query. However, our paper considers the problem in a more general scenario from the perspective of derivatives.

7 Conclusion

In this paper, we study the problem of auditing the ranking on graphs, where we aim to find the most influential graph elements (e.g., edges, nodes, subgraphs) w.r.t. graph ranking results. We formally define the Graph Ranking Auditing Problem by measuring the influence of each graph element as the rate of change in certain loss functions defined over the ranking vector and formulate it as an optimization problem. We extend the problem to more general scenarios where different types of network structures and ranking algorithms can be audited by our proposed algorithms. We propose a family of fast approximation algorithms, named AURORA, with $(1 - 1/e)$ approximation ratio and a linear complexity in both time and space. The extensive experimental evaluations on more than ten datasets across three network structures demonstrate that the proposed AURORA algorithms are able to identify influential graph elements and scale linearly on large graphs. The algorithms outperform baseline methods in all the cases on plain graphs and are able to compete with baseline methods on Network of Networks and attributed networks. In our case studies, the algorithms are able to provide sense-making results in different scenarios.

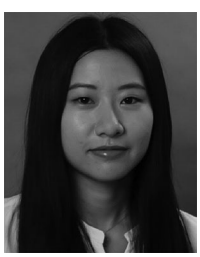
Acknowledgments

The faculty and students were supported by NSF (1947135, 1715385, and 1939725).

References

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford InfoLab, Stanford, CA, Tech. Rep. 1999–66, Nov. 1999. [Online]. Available: http://ilpubs.stanford.edu:8090/422/

[2] M. Gori and A. Pucci, "ItemRank: A random-walk based scoring algorithm for recommender engines," in *Proc. 20th Int. Joint Conf. Artif. Intell.*, 2007, pp. 2766–2771. [Online]. Available: http://dl.acm.org/citation.cfm?id=1625275.1625720
[3] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "Twitterrank: Finding topic-sensitive influential twitterers," in *Proc. 3rd ACM Int. Conf. Web Search Data Mining*, 2010, pp. 261–270. [Online]. Available: http://doi.acm.org/10.1145/1718487.1718520
[4] R. Singh, J. Xu, and B. Berger, "Pairwise global alignment of protein interaction networks by matching neighborhood topology," in *Proc. Annu. Int. Conf. Res. Comput. Mol. Biol.*, 2007, pp. 16–31.
[5] J. J. Crofts and D. J. Higham, "Googling the brain: Discovering hierarchical and asymmetric network structures, with applications in neuroscience," *Internet Math.*, vol. 7, no. 4, pp. 233–254, 2011.
[6] F. Radicchi, "Who is the best player ever? a complex network analysis of the history of professional tennis," *PloS One*, vol. 6, no. 2, 2011, Art. no. e17249.
[7] A. Das Sarma, A. R. Molla, G. Pandurangan, and E. Upfal, "Fast distributed pagerank computation," *Theor. Comput. Sci.*, vol. 561, pp. 113–121, Jan. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.tcs.2014.04.003
[8] C. Borgs, M. Brautbar, J. Chayes, and S.-H. Teng, "A sublinear time algorithm for pagerank computations," in *Proc. Int. Workshop Algorithms Models Web-Graph*, 2012, pp. 41–53.
[9] T. H. Haveliwala, "Topic-sensitive PageRank," in *Proc. 11th Int. Conf. World Wide Web*, 2002, pp. 517–526.
[10] H. Tong, C. Faloutsos, and J.-Y. Pan, "Fast random walk with restart and its applications," in *Proc. 6th Int. Conf. Data Mining*, 2006, pp. 613–622.
[11] L. Li, Y. Yao, J. Tang, W. Fan, and H. Tong, "Quint: On query-specific optimal networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 985–994.
[12] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1885–1894. [Online]. Available: http://dl.acm.org/citation.cfm?id=3305381.3305576
[13] J. Ni, H. Tong, W. Fan, and X. Zhang, "Inside the atoms: Ranking on a network of networks," in *Proce. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 1356–1365.
[14] R. Pienta, A. Tamersoy, H. Tong, and D. H. Chau, "Mage: Matching approximate patterns in richly-attributed graphs," in *Proc. IEEE Int. Conf. Big Data*, 2014, pp. 585–590.
[15] H. Whitney, "Congruent graphs and the connectivity of graphs," *Amer. J. Math.*, vol. 54, no. 1, pp. 150–168, 1932. [Online]. Available: http://www.jstor.org/stable/2371086
[16] W. W. Zachary, "An information flow model for conflict and fission in small groups," *J. Anthropol. Res.*, vol. 33, no. 4, pp. 452–473, 1977.
[17] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations," *Behavioral Ecol. Sociobiol.*, vol. 54, no. 4, pp. 396–405, 2003.
[18] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2010, pp. 1361–1370.
[19] L. Takac and M. Zabovsky, "Data analysis in public social networks," in *Proc. Int. Sci. Conf. Int. Workshop Present Day Trends Innovations*, 2012.
[20] L. Li, H. Tong, N. Cao, K. Ehrlich, Y.-R. Lin, and N. Buchler, "Replacing the irreplaceable: Fast algorithms for team member recommendation," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 636–646.
[21] M. Ley, "The DBLP computer science bibliography: Evolution, research issues, perspectives," in *Proc. Int. Symp. String Process. Inf. Retrieval*, 2002, pp. 1–10.
[22] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: Densification laws, shrinking diameters and possible explanations," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2005, pp. 177–187.
[23] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: Extraction and mining of academic social networks," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 990–998.
[24] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-MAT: A recursive model for graph mining," in *Proc. SIAM Int. Conf. Data Mining*, 2004, pp. 442–446.
[25] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of Internet portals with machine learning," *Inf. Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.
[26] D. E. Knuth, *The Stanford GraphBase: A Platform for Combinatorial Computing*, vol. 56, New York, NY, USA: ACM, 1993.
[27] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discovery from Data*, vol. 1, no. 1, 2007, Art. no. 2.
[28] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999.
[29] A. Y. Ng, A. X. Zheng, and M. I. Jordan, "Stable algorithms for link analysis," in *Proc. 24th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2001, pp. 258–266.
[30] C. Ding, X. He, P. Husbands, H. Zha, and H. Simon, "PageRank, hits and a unified framework for link analysis," in *Proc. SIAM Int. Conf. Data Mining*, 2003, pp. 249–253.
[31] G. Jeh and J. Widom, "SimRank: A measure of structural-context similarity," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2002, pp. 538–543.
[32] R. Albert, H. Jeong, and A.-L. Barabási, "Error and attack tolerance of complex networks," *Nature*, vol. 406, no. 6794, 2000, Art. no. 378.
[33] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han, "Attack vulnerability of complex networks," *Phys. Rev. E*, vol. 65, no. 5, 2002, Art. no. 056109.
[34] P. Boldi, M. Rosa, and S. Vigna, "Robustness of social and web graphs to node removal," *Social Netw. Anal. Mining*, vol. 3, no. 4, pp. 829–842, 2013.
[35] C. Martin and P. Niemeyer, "Comparing the sensitivity of social networks, web graphs, and random graphs with respect to vertex removal," in *Proc. 11th Int. Conf. Signal-Image Technol. Internet-Based Syst.*, 2015, pp. 460–467.
[36] H. Dai *et al.*, "Adversarial attack on graph structured data," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1123–1132.
[37] X. Wang, M. Cheng, J. Eaton, C.-J. Hsieh, and F. Wu, "Attack graph convolutional networks by adding fake nodes," 2018, *arXiv: 1810.10751*.
[38] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 2847–2856.
[39] M. Sun *et al.*, "Data poisoning attack against unsupervised node embedding methods," 2018, *arXiv: 1810.12881*.
[40] J. Chen, Z. Shi, Y. Wu, X. Xu, and H. Zheng, "Link prediction adversarial attack," 2018, *arXiv: 1810.01110*.
[41] K. Zhou, T. P. Michalak, T. Rahwan, M. Waniek, and Y. Vorobeychik, "Adversarial link prediction in social networks," 2018, *arXiv: 1809.08368*.
[42] J. Kang, S. Freitas, H. Yu, Y. Xia, N. Cao, and H. Tong, "X-rank: Explainable ranking in complex multi-layered networks," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 1959–1962.
[43] R. Varadarajan, V. Hristidis, and L. Raschid, "Explaining and reformulating authority flow queries," in *Proc. IEEE 24th Int. Conf. Data Eng.*, 2008, pp. 883–892.



Meijia Wang received the BS degree in economics, majored in financial mathematics from the Southern University of Science and Technology, China, and the MS degree in statistics from Arizona State University. She is currently working toward the PhD degree in statistics from the School of Mathematical and Statistical Sciences, Arizona State University. Her research interests include machine learning, graph mining, and bayesian statistics.

Jian Kang received the master's degree in computer science from the University of Virginia, and the BEng degree in communications engineering from the Beijing University of Posts and Telecommunications. He is currently working toward the PhD degree in computer science in the Department of Computer Science, University of Illinois at Urbana-Champaign. His current research interests include large-scale data mining and machine learning, especially graph mining.

Nan Cao received the PhD degree from The Hong Kong University of Science and Technology. He is a professor at TongJi University in China, with the joint appointment at both College of Design and Innovation and College of Software Engineering. He is the founding director of Tongji Intelligent Big Data Visualisation Lab (iDVx Lab). He is also an adjunct professor at NYU ShangHai and NYU Tandon. Before joining Tongji and NYU Shanghai, he was a research staff member at IBM T.J. Watson Research Center and he has worked for IBM for 10 years during 2005–2015.

Yinglong Xia received the BS degree from the University of Electronic Sciences and Technology of China, in 2003, the MS degree in machine learning from Tsinghua University, in 2006, and the PhD degree in computer science from the University of Southern California, in 2010. He is an applied research scientist at Facebook AI, working on various AI platform techniques and its applications, especially those focus on graph-based data processing and learning. He was a chief architect in Futurewei Technologies, Santa Clara, CA, leading a global team of scientists and engineers to build Big Data Analytics Platforms. Before that, he was a technical lead of graph database and reasoning frameworks and research staff member in the IBM T.J. Watson Research Center.

Wei Fan received the PhD degree in computer science from Columbia University, in 2001. He is currently the executive director of Tecent Medical AI Lab in Sunnyvale, CA. His main research interests and experiences are in various areas of data mining and database systems, such as deep learning, stream computing, high performance computing, extremely skewed distribution, cost-sensitive learning, risk analysis, ensemble methods, easy-to-use nonparametric methods, graph mining, predictive feature discovery, feature selection, sample selection bias, transfer learning, time series analysis, bioinformatics, social network analysis, novel applications, and commercial data mining systems.

Hanghang Tong received the MSc and PhD degrees in machine learning from Carnegie Mellon University, in 2008 and 2009. He is currently an associate professor with the Department of Computer Science, University of Illinois at Urbana-Champaign. Before that he was an associate professor at the School of Computing, Informatics, and Decision Systems Engineering (CIDSE), Arizona State University. His research interest is in large scale data mining for graphs and multimedia. He has received several awards, including SDM/IBM Early Career Data Mining Research Award (2018), NSF CAREER Award (2017), ICDM 10-Year Highest Impact Paper Award (2015), four best paper awards (TUP'14, CIKM'12, SDM'08, and ICDM'06), seven 'bests of conference', one best demo, honorable mention (SIGMOD'17), and one best demo candidate, second place (CIKM'17). He has published more than 100 refereed articles. He is the editor-in-chief of SIGKDD Explorations (ACM), an action editor of *Data Mining and Knowledge Discovery (Springer), and an associate editor of Knowledge and Information Systems (Springer) and Neurocomputing Journal (Elsevier)*; and has served as a program committee member in multiple data mining, database, and artificial intelligence venues (e.g., SIGKDD, SIGMOD, AAAI, WWW, CIKM, etc.).