Inside the Atoms: Ranking on a Network of Networks

Jingchao Ni 1, Hanghang Tong 2, Wei Fan 1, and Xiang Zhang 1
1Department of Electrical Engineering and Computer Science, Case Western Reserve University
2School of Computing, Informatics, Decision Systems Engineering, Arizona State University

ABSTRACT

Networks are prevalent and have posed many fascinating research questions. How can we spot similar users, e.g., virtual identical twins, in Cleveland for a New Yorker? Given a query disease, how can we prioritize its candidate genes by incorporating the tissuespecific protein interaction networks of those similar diseases? In most, if not all, of the existing network ranking methods, the nodes are the ranking objects with the finest granularity.

In this paper, we propose a new network data model, a Network of Networks (NoN), where each node of the main network itself can be further represented as another (domain-specific) network. This new data model enables to compare the nodes in a broader context and rank them at a finer granularity. Moreover, such an NoN model enables much more efficient search when the ranking targets reside in a certain domain-specific network. We formulate ranking on NoN as a regularized optimization problem; propose efficient algorithms and provide theoretical analysis, such as optimality, convergence, complexity and equivalence. Extensive experimental evaluations demonstrate the effectiveness and the efficiency of our methods.

Categories and Subject Descriptors
H.2.8 [Database Management]: Database Applications—Data mining

Keywords
Network of Networks; Ranking; Query

1. INTRODUCTION

How can we spot similar users, e.g., virtual identical twins, in Cleveland for a New Yorker? Given a disease, how can we prioritize its candidate genes by incorporating the tissue-specific protein interaction networks of those similar diseases? Among all the researchers in the area of bioinformatics, who is most likely to collaborate with a given data miner in the near future? In these applications, networks (or graphs 1 ) provide a natural data model to capture the relationship among different objects.

In the past decade, many graph ranking algorithms have been proposed. Please see Section 6 for a brief review. Despite their own success, a common limitation of these methods is that ranking stops at the node (atom) level. That is, the nodes are the ranking objects with the finest granularity in these existing ranking algorithms.

In this paper, we take one step further and propose a new data model — a Network of Networks (NoN). In this NoN model, each node of the main network can be represented as another network. Let us elaborate using the example in Figure 1. Here, the dashed network represents a geo-proximity network (the main network), where the nodes are five different cities and links indicate the geo-proximity among different cities. Each node of this main network is further represented as a social network (the domain-specific network), where nodes are users and links denote their friendship. Collectively, we call this structure as a (geo-proximity) network of (social) networks.

The main advantage of this new NoN model is that it enables to compare the nodes in a broader context and rank them at a finer granularity. Take Figure 1 as the example, using the existing network models, we could either rank the geo-centrality of the cities, or the popularity of the users within each domain-specific network. On the other hand, by the NoN model, we will be able to rank all the users by considering both their popularity within each domain-specific network as well as the geo-centrality of the locations they belong to. We refer to this setting as CROSSRANK. Moreover, we

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD ’14, August 24–27, 2014, New York, NY, USA.
Copyright 2014 ACM 978-1-4503-2956-9/14/08 . . $15.00.
http://dx.doi.org/10.1145/2623330.2623643.

1In this paper, we use network and graph interchangeably.
might be interested in finding who are the most similar from New York City for Jon from Cleveland, i.e., who are the virtual identity twins of Jon in New York City. In this case, the ranking targets reside in a certain domain-specific network. We refer to this setting as CROSSQUERY. The NoN model will enable much more efficient search for this problem.

The NoN model can be observed in many real-world applications. As another example, consider a NoN, where each node in the main network is a research area, and the domain-specific networks are the co-author networks in different areas. By the NoN model, we can rank authors according to their contributions to multiple areas using CROSSRANK; and find future collaborators for a researcher in a different area using CROSSQUERY.

The main contributions of this paper can be summarized as:

- **New Data Model.** We propose a novel network data model, a network of networks, which enables network ranking to go beyond the atom/node level (Section 2).

- **Algorithms and Analysis.** We propose two novel ranking algorithms (CROSSRANK and CROSSQUERY) in NoN and analyze their optimality, convergence, complexity and relationship with existing ranking algorithms (Section 3 and 4).

- **Empirical Evaluations.** We perform extensive experiments on real datasets to validate the effectiveness and efficiency of the proposed algorithms (Section 5).

## 2. PROBLEM DEFINITIONS

Table 1 lists the main symbols used throughout this paper. We use the bold capital letters to denote matrices (e.g., G, A, etc.), the bold lower cases for vectors (e.g., r) and calligraphic letters for sets (e.g., A). In this paper, we consider weighted undirected networks. We denote the networks by their corresponding weighted adjacency matrices. For example, a network with g nodes can be denoted by a g×g adjacency matrix G. The rows/columns of the matrix G represent the nodes of the network and its non-zero element G(i, j) measures the link strength from node i to node j. With the above notations, a network of networks (NoN) can be formally defined as follows.

**Definition 1.** Network of Networks (NoN). Given a g×g main network G, a set of g domain-specific networks A = {A_1, ..., A_g} and a one-to-one mapping function θ, which maps each node in the main network G to a domain-specific network, a Network of Networks (NoN) is defined as the triplet R = < G, A, θ >. Nodes in the main network are referred to as main nodes, nodes in the domain-specific networks are called domain nodes. Each main node represents a domain-specific network through the mapping function θ. In addition, we represent the nodes in each domain-specific network as V_i (i = 1, ..., g). We define I_{ij} as the common nodes between A_i and A_j, i.e., I_{ij} = V_i ∩ V_j.

Take Figure 1 as an example, the dashed network is the main network G, which has five main nodes, including NYC, Pittsburgh, Cleveland, Philadelphia, DC. Each of these main nodes is mapped to a domain-specific network (the solid network). For example, the main node Pittsburgh is further represented as a domain-specific network with three nodes, including Jim, Ben, Bob.

Given an NoN R = < G, A, θ >, a fundamental task is to compare and rank the nodes in this network. For instance, we might be interested in ranking the importance/popularity of the individuals in each domain-specific network. Further more, given an individual, e.g., Jim from Pittsburgh, we may also want to find the similar users from NYC as well as other cities.

We introduce a query vector e_i (i = 1, ..., g) for each domain-specific network, which is an n_i×1 nonnegative vector (n_i represents the number of nodes in the domain-specific network). If we are given a query node of interest from a domain-specific network (e.g., Jim from Pittsburgh), we set its corresponding entry e_i to 1, and all other entries to 0. Otherwise, all the entries in e_i are set to 1/7. Formally, we define the CROSSRANK problem as follows.

**Problem 1. CROSSRANK**

**Given:** (1) an NoN R = < G, A, θ >, and (2) the query vectors e_i (i = 1, ..., g);

**Find:** ranking vectors r_i for the nodes in the domain-specific networks A_i (i = 1, ..., g).

In the above definition, when all the query vectors are uniform, i.e., no query node exists, we essentially want to measure and rank the global importance of all the domain nodes within each domain-specific network. On the other hand, if a query node is given, we essentially want to measure the proximity (i.e., relevance) from the query node to all the domain nodes within each domain-specific network. Note that in this latter case, sometimes, we might be interested in ranking domain nodes from a particular domain-specific network. For instance, given Jim from Pittsburgh, we might want to find out the top-3 similar users from NYC. We call this special case of CROSSRANK as CROSSQUERY, which is formally defined as follows.

**Problem 2. CROSSQUERY**

**Given:** (1) an NoN R = < G, A, θ >, (2) a query node of interest from a source domain-specific network A_s, (3) a target domain-specific network A_t, and (4) an integer k;

**Find:** the top-k most relevant nodes from the target domain-specific network A_t w.r.t. the query node.

In the next two sections, we give our solutions for CROSSRANK and CROSSQUERY, respectively.

## 3. CROSSRANK

In this section, we present our solution to CROSSRANK. We start with formulating it as a regularized optimization problem, then present an effective algorithm to solve it, followed by some theoretical analysis.
3.1 Optimization Formulation

The basic idea of our method is to formulate CROSSRANK as a regularized optimization problem by encoding the following three types of constraints. (1) **Within-network smoothness**: for a given domain-specific network \( A_i \), the ranking scores among the neighboring nodes should be somehow smooth. That is, for any two adjacent nodes \( x, y \) on \( A_i \), we want to minimize the difference of the rank scores between them, i.e., \( r_i(x) - r_i(y) \). (2) **Query preference**: the ranking vector \( r_i \) should also reflect the preference of the specific query node of interest from the corresponding domain-specific network. That is, we want to minimize the difference between the ranking vector \( r_i \) and the corresponding query vector \( e_i \). (3) **Cross-network consistency**: for a pair of domain-specific networks \( A_i \) and \( A_j \) that are connected with each other by the main network (i.e., \( G(i, j) > 0 \)), they may share some common nodes, i.e., \( I_{ij} = V_i \cap V_j \) is non-empty. In this case, those common nodes will receive multiple ranking scores (one from each ranking vector). Intuitively, the ranking scores for the same node should be consistent across highly similar domain-specific networks. In other words, we want to minimize the difference between \( r_i(I_{ij}) \) and \( r_j(I_{ij}) \). Here \( r_i(I_{ij}) \) can be viewed as the projection of the entire ranking vector \( r_i \) on the common node set \( I_{ij} \).

Putting everything together, we have the following regularized objective function \( J(r_1, \ldots, r_g) \). The optimal ranking vectors \( r_i \) (\( i \in \{1, \ldots, g\} \)) are those minimize the objective function.

\[
J(r_1, \ldots, r_g) = c \sum_{i=1}^{g} \sum_{j=1}^{n} (r_i(I_{ij}) - \hat{A}_i r_i) + (1 - c) \sum_{i=1}^{g} \|r_i - e_i\|_F^2 \quad \text{within-network smoothness}
\]

\[
+ \alpha \sum_{i=1}^{g} \sum_{j=1}^{n} \frac{\|r_i(I_{ij})\|_F}{\sqrt{d_{m}(j)}} \frac{\|r_j(I_{ij})\|_F}{\sqrt{d_{m}(j)}} \quad \text{query preference}
\]

\[
+ a \sum_{i=1}^{g} \sum_{j=1}^{n} \frac{\|r_i(I_{ij})\|_F}{\sqrt{d_{m}(j)}} \frac{\|r_j(I_{ij})\|_F}{\sqrt{d_{m}(j)}} \quad \text{cross-network consistency}
\]

where \( \| \cdot \|_F \) denotes the Frobenius norm; \( \hat{A}_i \) is the symmetrically normalized adjacency matrix of \( A_i \); and \( a \) and \( c \) are two regularization parameters. Note that the length of different ranking vectors could be different and all entries of a given ranking vector \( r_i \) add up to 1. To make the ranking scores comparable with each other for the common nodes, we normalize \( r_i(I_{ij}) \) and \( r_j(I_{ij}) \) by their corresponding degrees in the main networks, respectively.

**Discussions.** If we ignore the third term (cross-network consistency) in Eq. (1), we can show that the objective function \( J(r_1, \ldots, r_g) \) can be decoupled into \( g \) independent terms. Each of these terms becomes the same objective function of a popular ranking method on homogeneous network - random walk with restart (also called manifold ranking, the personalized pagerank, etc.) [33, 26]. From this perspective, what we essentially aim to do in Eq. (1) is to solve multiple, inter-correlated such ranking problems simultaneously.

3.2 Optimization Solutions

We first present an equivalent formulation of Eq. (1). Denote the aggregated ranking vector \( r = (r'_1, \ldots, r'_g)' \), the aggregated query vector \( e = (e'_1, \ldots, e'_g)' \). Let the aggregated adjacency matrix \( \hat{A} = \text{diag}(\hat{A}_1, \ldots, \hat{A}_g) \) be a \( g \times g \) block diagonal matrix, \( O_j \) be an \( n \times n \) binary indicator matrix with \( O_j(x, y) = 1 \) if the \( x^{\text{th}} \) node in \( A_i \) is the \( y^{\text{th}} \) node in \( A_i \), i.e., common node. Then \( O \) is a block matrix whose \((i, j)^{\text{th}}\) block is \( O \). \( Y \) is non-singular. The \((i, j)^{\text{th}}\) block of \( D_y \) (i.e., \( Y \)’s degree matrix) equals \( d_{m}(r_i)I_{ik} \). Finally, we define \( X = D_{r}^{-\frac{1}{2}}(D_r - Y)D_r^{-\frac{1}{2}} \), \( \bar{Y} = D_{r}^{-\frac{1}{2}}YD_{r}^{-\frac{1}{2}} \).

With these definitions, the objective function in Eq. (1) can be re-written as the following matrix form

\[
J(r) = cr'(I_n - \hat{A})r + (1 - c)\|r - e\|_F^2 + 2ar'Xr
\]

where \( n = \sum_{i=1}^{g} n_i \) is the total number of nodes in all the domain-specific networks.

**Lemma 1.** Equation (1) and Equation (2) are equivalent.

**Proof.** We only need to prove that the third term in Eq. (1) is equal to that in Eq. (2).

By the definition of \( X \) and \( r \), we have

\[
r'Xr = r'1_n r - r'Yr = \sum_{i=1}^{g} r'_i 1_n r_i - \sum_{i=1}^{g} r'_i \bar{Y} r_j\]

where \( \bar{Y} \) is the \((i,j)\)th block of \( \bar{Y} \) of size \( n_i \times n_j \). Then

\[
r'Xr = \frac{1}{2} \left( \sum_{i=1}^{g} \left( \sum_{j=1}^{n} \frac{r'_i}{\sqrt{d_{m}(i)}} (D_{r_{ij}})_{ij} \frac{r_j}{\sqrt{d_{m}(j)}} \right) \right)
\]

\[
- 2 \sum_{i=1}^{g} \sum_{j=1}^{n} \left( \frac{r'_i}{\sqrt{d_{m}(i)}} G(i, j) O_j \frac{r_j}{\sqrt{d_{m}(j)}} + \frac{r'_j}{\sqrt{d_{m}(j)}} (D_{r_{ij}})_{ij} \frac{r_i}{\sqrt{d_{m}(i)}} \right)
\]

\[
+ \sum_{j=1}^{n} \frac{r'_j}{\sqrt{d_{m}(j)}} (D_{r_{ij}})_{ij} \frac{r_j}{\sqrt{d_{m}(j)}}
\]

where \( (D_{r_{ij}})_{ij} \) and \( O_j \) are block matrices representing \((i, j)\)th block of \( D_r \), and \( O_j \) is the \((i, j)\)th block of \( Y \). Thus

\[
r'Xr = \frac{1}{2} \left( \sum_{i=1}^{g} \sum_{j=1}^{n} \frac{r'_i}{\sqrt{d_{m}(i)}} (D_{r_{ij}})_{ij} \frac{r_j}{\sqrt{d_{m}(j)}} + \frac{r'_j}{\sqrt{d_{m}(j)}} (D_{r_{ij}})_{ij} \frac{r_i}{\sqrt{d_{m}(i)}} \right)
\]

\[
- 2 \sum_{i=1}^{g} \sum_{j=1}^{n} \left( \frac{r'_i}{\sqrt{d_{m}(i)}} G(i, j) O_j \frac{r_j}{\sqrt{d_{m}(j)}} + \frac{r'_j}{\sqrt{d_{m}(j)}} (D_{r_{ij}})_{ij} \frac{r_i}{\sqrt{d_{m}(i)}} \right)
\]

\[
+ \sum_{j=1}^{n} \frac{r'_j}{\sqrt{d_{m}(j)}} (D_{r_{ij}})_{ij} \frac{r_j}{\sqrt{d_{m}(j)}}
\]

and

\[
\sum_{i=1}^{g} \sum_{j=1}^{n} \frac{r'_i}{\sqrt{d_{m}(i)}} (D_{r_{ij}})_{ij} \frac{r_j}{\sqrt{d_{m}(j)}} = 2 \sum_{i=1}^{g} \sum_{j=1}^{n} \frac{r'_i}{\sqrt{d_{m}(i)}} (D_{r_{ij}})_{ij} \frac{r_j}{\sqrt{d_{m}(j)}}
\]

\[
\sum_{i=1}^{g} \sum_{j=1}^{n} \frac{r'_i}{\sqrt{d_{m}(i)}} (D_{r_{ij}})_{ij} \frac{r_j}{\sqrt{d_{m}(j)}} = 0
\]

This means \( D_r \) does not affect the ranking results. Therefore, we have that

\[
r'Xr = \frac{1}{2} \sum_{i=1}^{g} \sum_{j=1}^{n} G(i, j) \left( \frac{r'_i(I_{ij})}{\sqrt{d_{m}(i)}} \frac{r_j(I_{ij})}{\sqrt{d_{m}(j)}} - 2 \frac{r'_i(I_{ij})}{\sqrt{d_{m}(i)}} \frac{r_j(I_{ij})}{\sqrt{d_{m}(j)}} \right)
\]

\[
+ \frac{r'_j(I_{ij})}{\sqrt{d_{m}(j)}} \frac{r_i(I_{ij})}{\sqrt{d_{m}(i)}}
\]

\[
= \frac{1}{2} \sum_{i=1}^{g} \sum_{j=1}^{n} G(i, j) \|r'_i(I_{ij}) - r_j(I_{ij})\|_F^2
\]

This completes the proof. \(\square\)
Algorithm 1: CROSSRANK

**Input:** (1) a network of networks $R = \langle G, \mathcal{A}, \theta \rangle$; (2) the query vectors $e_i$ ($i = 1, ..., g$); and (3) the parameters $a$ and $c$.

**Output:** the ranking vectors $r_i$, ($i = 1, ..., g$)

1. **Offline-computation:** Construct $\tilde{A}$ and $\tilde{Y}$ from $R$;
2. **Online-ranking:**
   3. Construct the aggregated query vector $e = (e'_1, ..., e'_g)'$;
   4. Initialize the aggregated ranking vector $r = e$;
5. **while not convergence**
6. \hspace{1em} Update: $r \leftarrow (\frac{c}{1 + 2a} \tilde{A} + \frac{2a}{1 + 2a} \tilde{Y})r + \frac{1 - c}{1 + 2a} e$;
7. **end**
8. return the ranking vectors $r_1, ..., r_g$ based on $r$

From Eq. (2), it can be seen that the objective function $J$ is a quadratic function of the ranking vector $r$. We have

$$\frac{\partial J}{\partial r} = 2(1 + 2a)I_n - c\tilde{A} - 2a\tilde{Y}r - 2(1 - c)e$$

If we set $r = r - \eta \frac{\partial J}{\partial r}$, where $\eta = \frac{1}{1 + 2a}$, we have

$$r = (\frac{c}{1 + 2a} \tilde{A} + \frac{2a}{1 + 2a} \tilde{Y})r + \frac{1 - c}{1 + 2a} e \quad (3)$$

Based on Eq. (3), we have a fixed-point approach to compute the optimal ranking vector $r$ that minimizes the objective function $J(r)$ as illustrated in Algorithm 1.

### 3.3 Proofs and Analysis

In this subsection, we analyze Algorithm 1 in terms of its convergence, optimality, complexity and discuss its relationship with existing ranking algorithms.

We start with the convergence of Algorithm 1, which is summarized in Theorem 1. It says that the proposed CROSSRANK algorithm converges to its closed-form solution.

**Theorem 1. Convergence of CROSSRANK.** Algorithm 1 converges to the closed-form solution: $r = (I_n - \frac{c}{1 + 2a} \tilde{A} - \frac{2a}{1 + 2a} \tilde{Y})^{-1} \frac{1 - c}{1 + 2a} e$.

**Proof.** First, the closed-form solution can be obtained by solving $A_r = 0$. Then let $M = \frac{1}{1 + 2a} \tilde{A} + \frac{2a}{1 + 2a} \tilde{Y}$. Eq. (3) becomes $r = Mr + \frac{1 - c}{1 + 2a} e$. Next, we show that the eigenvalues of $M$ are in $(-1, 1]$.

Since $\tilde{A}$ and $\tilde{Y}$ are similar to the stochastic matrices $AD_\mathcal{A}$ and $YD_\mathcal{Y}$, respectively, both $\tilde{A}$ and $\tilde{Y}$ have eigenvalues within $[-1, 1]$.

One result of the Weyl’s inequality theorem [1] states that for matrices $\tilde{H}, \tilde{H}, P \in \mathcal{H}_n$, where $\mathcal{H}_n$ is the set of $n \times n$ Hermitian matrices, if $\tilde{H} = \tilde{H} + P$ and their eigenvalues are arranged in non-increasing orders, i.e., $\lambda_i(\tilde{H}) \geq \cdots \geq \lambda_1(\tilde{H})$, $\lambda_i(\tilde{H}) \geq \cdots \geq \lambda_1(\tilde{H})$, then the following inequalities hold:

$$\lambda_i(M) \leq \lambda_i(\tilde{H}) - \lambda(\tilde{P}) \leq \lambda_i(P), \forall i = 1, ..., n$$

Since $\tilde{A}, \tilde{Y}, M \in \mathcal{H}_n$, and $M = \frac{1}{1 + 2a} \tilde{A} + \frac{2a}{1 + 2a} \tilde{Y}$, we have

$$\lambda_i(M) \leq \lambda_i(\tilde{A}) + \frac{2a}{1 + 2a} \lambda_i(\tilde{Y})$$

which means the eigenvalues of $M$ are in the range of $[-\frac{c - 2a}{1 + 2a}, \frac{c + 2a}{1 + 2a}]$. Since $0 < c < 1$, the eigenvalues of $M$ are in $(-1, 1]$. Based on this property, we can show the convergence of the fixed-point approach. Without loss of generality, let $r(0) = e$, and $t$ be the iteration index ($t \geq 1$). According to Eq. (3), we have

$$r(t) = Me + \sum_{i=0}^{t-1} M^i(1 - c) e \quad \frac{1 - c}{1 + 2a}$$

Since the eigenvalues of $M$ are all in $(-1, 1)$, we have

$$\lim_{t \to \infty} M^i = 0,$$

and

$$\lim_{t \to \infty} \sum_{i=0}^{t-1} M^i = (I_n - M)^{-1}$$

Therefore

$$r = \lim_{t \to \infty}r(t) = (I_n - M)^{-1}(1 - c) e \quad \frac{1 - c}{1 + 2a}$$

which is the closed-form solution.

Next, we show in Lemma 2 that Algorithm 1 finds the global optimal solution of the objective function $J(r)$ defined in Eq. (2).

**Lemma 2. Optimality of CROSSRANK.** At convergence, Algorithm 1 finds the global minimal solution of $J(r)$ defined in Eq. (2).

**Proof.** This can be proved by showing that the function in Eq. (2) is convex. The Hessian matrix of Eq. (2) is $\nabla^2 J = 2((1 + 2a)I_n - c\tilde{A} - 2a\tilde{Y})$. Following the similar idea as in the proof of Theorem 1, we have that the eigenvalues of $\nabla^2 J$ are no less than $2(1 - c)$. Since $0 < c < 1$, $\nabla^2 J$ is positive-definite. Therefore, Eq. (2) is convex.

The complexity of Algorithm 1 is summarized in Lemma 3, which says that it is linear w.r.t. the overall number of nodes and edges in NoN in terms of both space and time cost.

**Lemma 3. Complexity of CROSSRANK.** The time complexity of Algorithm 1 is $O(T^*(m + ng))$ and its space complexity is $O(m + gn)$, where $m = \sum_{i=1}^{g} m_i$, $n = \sum_{i=1}^{g} n_i$, and $T^*$ is the total iteration number.

**Proof.** In the preprocessing stage, we need to construct $\tilde{A}$ and $\tilde{Y}$. The construction time of $\tilde{A}$ is $O(mn)$, since there are $O(mn)$ nonzero entries in $\tilde{A}$. The construction time of $\tilde{Y}$ is $O(ng)$, since the nonzero entries of $\tilde{Y}$ are pairwise intersection of node lists of all the domain-specific networks, i.e., $\sum_{i=1}^{g} gn_i = gn$. Therefore, preprocessing cost is $O(m + gn)$. The online-computation cost depends on the fixed-point updating step in Eq. (3). Since there are $O(m + gn)$ nonzero entries in $\frac{1}{1 + 2a} \tilde{A} + \frac{2a}{1 + 2a} \tilde{Y}$, the online-computation cost is $O(T^*(m + gn))$. In Algorithm 1, we need to store $\tilde{A}$ and $\tilde{Y}$, with space complexity $O(m + gn)$. Usually $g$ is much smaller than $n$, we can regard time and space complexities as $O(T^*(m + n))$ and $O(m + n)$ respectively.

Finally, we show the relationship between the proposed CROSSRANK algorithm and random walk with restart (RWR).

**Lemma 4. Relationship between CROSSRANK and RWR.** CROSSRANK fixed-point approach is equivalent to random walk with restart with the transition matrix $W = \frac{c}{1 + 2a} \tilde{A} + \frac{2a}{1 + 2a} YD_\mathcal{Y}$ if $\tilde{A}$ is stochastic.

**Proof.** Let $\tilde{c} = \frac{c}{1 + 2a}$ and $W = \frac{c}{1 + 2a} \tilde{A} + \frac{2a}{1 + 2a} \tilde{Y}$, we can transform Eq. (3) to

$$r = \tilde{c} Wr + (1 - \tilde{c}) e \quad (4)$$

and the closed-form solution of Eq. (2) to

$$r = (I_n - \tilde{c} W)^{-1}(1 - \tilde{c}) e \quad (5)$$
These formulations have a RWR form except that \( W \) is not stochastic, since \( \tilde{A} \) and \( Y \) are symmetrically normalized. If we denote 
\[
\hat{W} = \frac{c}{\sigma^2} \tilde{A} + \frac{2c}{\sigma^2} YD^{-1}Y^T,
\]
we can further transform the closed-form solution to 
\[
\hat{r} = (I_n - \frac{c}{1+2\sigma} \tilde{A} - \frac{2c}{1+2\sigma} YD^{-1}Y^T)^{-1} \left[ 1 - \frac{c}{1+2\sigma} e \right] = D^{-1}_Y(I_n - \frac{c}{1+2\sigma} \tilde{A} - \frac{2c}{1+2\sigma} YD^{-1}Y^T)^{-1} \left[ 1 - \frac{c}{1+2\sigma} e \right] \tag{6}
\]
\[
= D^{-1}_Y(I_n - \frac{c}{1+2\sigma} \hat{W} - \frac{2c}{1+2\sigma} YD^{-1}Y^T)^{-1} \left[ 1 - \frac{c}{1+2\sigma} e \right]
\]
In the above equation, \( D^{-1}_Y(I_n - \frac{c}{1+2\sigma} \hat{W} - \frac{2c}{1+2\sigma} YD^{-1}Y^T)^{-1} \) is a diagonal block matrix, of which each diagonal block can be represented as \( (D_Y)^{-1}(A_Y, (D_Y)^{-1} = (A_Y, \hat{A}) \) and \( \hat{A} \) are the diagonal block of \( D_Y \) and \( \hat{A} \) respectively. Since the diagonal values of \( (D_Y)^{-1} \) equal to \( d_Y(i), (D_Y)^{-1}(A_Y, (D_Y)^{-1} = (A_Y, \hat{A}) \) and \( \hat{A} \) across domain-specific networks (note this process converges since \( \hat{W} \) and \( \hat{W} \) are similar). It gives the same ranking results as Eq. (4), since multiplying \( D^{-1}_Y \) on the ranking scores \( \hat{r} \) does not affect the relative ranking order within the domain-specific networks. If \( \hat{A} \) is stochastic, \( \hat{W} \) is stochastic and \( \hat{r} = \hat{c}W + (1 - \hat{c})e \) becomes a stochastic process. This completes the proof. □

4. CROSSQUERY

In this section, we address CROSSQUERY. Note that CROSSQUERY is a special case of CROSSRANK, by (1) requiring the query node being from a source domain-specific network \( A_s \); (2) restricting the query results within a target domain-specific network \( A_t \); and (3) searching for a set of \( k \) most relevant nodes from \( A_t \) (as opposed to demanding the exact ranking vectors). Thus, a default solution for CROSSQUERY is to run the CROSSRANK algorithm to get the ranking vector \( r_x \) for the target domain-specific network and return the top-\( k \) nodes with the highest ranking scores. In this section, we aim to further speed-up its computation.

4.1 CROSSQUERY-BASIC

According to Lemma 4, CROSSRANK has a RWR form on the integrated normalized matrix \( \hat{W} \). This not only reveals an interesting relationship between CROSSRANK and RWR, but also opens the door to take the advantage of the vast machinery of the existing scalable algorithms for RWR [25, 5, 6]. For example, we could modify the algorithm in [6] to solve CROSSQUERY since \( \lim_{t \to \infty}(\hat{c}W) = 0 \) (a prerequisite of this algorithm), which is summarized in Algorithm 2. This is an exact method that improves the basic power iteration method by estimating the node ranking scores using only a small number of iterations.

**Time Complexity.** Following the notations in Lemma 3, the overall time complexity of CROSSQUERY-BASIC is \( O(T(bf + h) + m + gn) \), where \( b \) is the average size of the neighborhood for each node, \( f \) and \( h \) are the average size of the sub-tree \( T^{(o)} \) and the selected nodes set \( S^{(o)} \) over all iterations respectively, \( T \) is the total number of iterations. Compared with the CROSSRANK method, whose time complexity is \( O(T'(m + gn)) \), where \( T' \) is the total number of iterations, Algorithm 2 is more efficient since in practice \( bf << m + gn, h << n \) and \( T << T' \).

**Algorithm 2: CROSSQUERY-BASIC (adapted from [6])**

**Input:** (1) matrices \( W, W_{max} \) (\( W_{max}(u) = \max_{v \in N(u)} W(u,v) \) where \( N(u) \) is the neighbourhood of \( u \); (2) node sets \( V_{t_1}, V_{t_2}, \ldots, V_{t_n} \); (3) nodes \( q, s, d \) and (4) parameters \( \epsilon, k \)

**Output:** Top \( k \) relevant nodes set \( \hat{S} \)

1. Set \( \epsilon(q) = 1, \epsilon = 0, \forall v \in (1, \ldots, g)s \), and the query vector \( \epsilon = (\epsilon'_1, \ldots, \epsilon'_s) \).
2. Initialize the random walk vector \( p = e \), the lower bound vector \( L^{(0)} = 0 \), and the upper bound vector \( U^{(0)} = 0 \).
3. Initialize the iteration number \( t = 0 \), Subtree \( T^{(0)} = \{q\}, \) the selected nodes set \( \hat{S}^{(0)} = \{q\} \), the threshold \( \sigma = -1 \);
4. while \( |S^{(t)}| > k \) do
5. Increase the iteration number \( t = t + 1 \);
6. Expand \( T^{(t)} = T^{(t-1)} \cup layer^{(t)} \);
7. Update \( p(T^{(t)}) = \hat{W}p(T^{(t)}) \);
8. Update lower bound: \( L^{(t)} = L^{(t-1)} + (1 - \epsilon) \hat{c} \hat{c} \hat{p}(S^{(t-1)}) \);
9. Update upper bound: \( U^{(t)} = U^{(t)} + \hat{c} \hat{c} \hat{p}(S^{(t-1)}) \);
10. Update threshold: \( \sigma = \hat{c} \hat{c} \hat{p}(S^{(t)}) \);
11. Select nodes from \( S^{(t)} \) using \( \epsilon' \) and \( \epsilon' \) thresholds
12. Update total score:
13. end
14. return \( \hat{S} = S^{(t)} \).

4.2 CROSSQUERY-Fast

Next, we propose a more efficient algorithm for CROSSQUERY. The basic idea is as follows. Let the corresponding main nodes for \( A_s \) and \( A_t \) be node \( s \) and node \( d \), respectively. Given \( s \) and \( d \), parts of the main network \( G \) might have little contributions to measure the relevance for the nodes from \( A_s \) w.r.t. the query node, and thus can be pruned with little impact on the final query accuracy. In the context of NoN, even if we can only prune a small portion of the main network, the computational efficiency can be significantly improved since all the corresponding domain-specific networks can be filtered out during the ranking process.

Let \( \epsilon(v) \) be the ranking score of any node \( v \) w.r.t. a query node \( q \) by RWR. It is well known that \( \epsilon(v) \) can be represented by the so-called inverse P-distance [9].

\[
\epsilon(v) = \sum_{p \epsilon(V_{t_1})} Prob(p)(1 - c)e^{h(p)}
\]

where \( p \in \{q \ldots v\} \) is a path from \( q \) to \( v \), and \( h(p) \) is the unweighted length of \( p \). For any path \( p : v_1 \to v_2 \to \ldots \to v_{|p|} \),
\[
Prob(p) = \prod_{i=1}^{|p|-1} Prob(v_i, v_{i+1}),
\]
where \( Prob(v_i, v_{i+1}) \) is the transition probability from \( v_i \) to \( v_{i+1} \). We call \( Prob(p) \) the transition probability of path \( p \).

Inspired by the relationship between CROSSRANK and RWR (as indicated by Eq. (6)), here we consider \( \hat{A} \) to be stochastic (i.e., column normalized) to guide the pruning process. Then we can express CROSSQUERY as a node \( v \) w.r.t. the query node \( q \) as

\[
\epsilon(v) = \sum_{p \epsilon(V_{t_1})} \frac{\epsilon^{d(v)}}{\epsilon^{d(d)}} Prob(p)(1 - \hat{c})e^{h(p)}
\]

The following lemma says that \( Prob(p), \forall p \in \{q \ldots v\}, \exists p' \in \{s \ldots d\} \) s.t. \( Prob(p) \leq Prob(p') \), where \( p' \) is a path con-
necting $s$ and $d$ in the main network. Here $\text{Prob}(p') = \prod_{i,j \in p'} \tilde{G}(i,j)$
where $e_{ij}$ is the edge connecting main nodes $i$ and $j$ in $G$.

**Proof.** For any edge $e_{ij} \in p$, let $i$ and $j$ be the main nodes that contain domain nodes $x$ and $y$ respectively. We have

$$\text{Prob}(p) = \prod_{i,j \in p} \frac{c}{c + 2a} \tilde{A}(x,y) \prod_{i,j \in p} \frac{2a}{c + 2a} \text{Prob}(\text{YD}^{i,j}_{\text{r}}(y,x))$$

$$\leq \prod_{i,j \in p} \text{Prob}(\text{YD}^{i,j}_{\text{r}}(y,x))$$

$$= \prod_{i,j \in p} \text{G}(i,j) \frac{\text{Prob}(p')}{d_{\text{ad}(i)}} = \text{Prob}(p')$$

where $(\text{YD}^{i,j}_{\text{r}}(y,x))$ is the $(i,j)^{th}$ block of $\text{YD}^{i,j}_{\text{r}}$, which is the transition matrix from domain-specific network $i$ to $j$. □

In the above lemma, $p'$ consists of edges in $p$ that go across domain-specific networks. Let $\varphi$ be the function that maps each $p$ to its corresponding $p'$, i.e., $\varphi(p) = p'$, then we have the following upper bound of the ranking score $r(v)$.

**Corollary 1. Upper Bound of $r(v)$**. The crossquery score of a node $v \in \mathcal{V}_L$ w.r.t. the query node $q$, i.e., $r(v)$, is upper bounded by

$$r(v) \leq \sum_{p \ni \varphi(v)} \sqrt{d_{\text{ad}(i)}} \text{Prob}(\varphi(p))(1 - \epsilon)^{d(p)}$$

**Proof.** This can be derived directly from Eq. (7) and Lemma 5. The detailed proof is omitted. □

From Corollary 1, we have that if $\text{Prob}(\varphi(p))$ is small, the contribution from the path $p$ to $r(v)$ will be small. Based on this, we can extract a subnetwork of the main network consisting of paths with large $\text{Prob}(\varphi(p))$.

For any $p$, let $\text{Prob}(\varphi(p)) = \frac{\sqrt{d_{\text{ad}(i)}}}{\sqrt{d_{\text{ad}(j)}}} \text{Prob}(\varphi(p)) = \prod_{i,j \in p} \frac{\text{G}(i,j)}{\sqrt{d_{\text{ad}(i)}} \sqrt{d_{\text{ad}(j)}}} \text{Prob}(\varphi(p)).$

We define the distance between node $i$ and node $j$ as

$$L_{ij} = -\log \left( \frac{\text{G}(i,j)}{\sqrt{d_{\text{ad}(i)}} \sqrt{d_{\text{ad}(j)}}} \right)$$

(8)

Then the upper bound of the ranking score $r(v)$ in Corollary 1 becomes

$$r(v) \leq \sum_{p \ni \varphi(v)} 10^{-\sum_{i,j \in p} L_{ij}} (1 - \epsilon)^{d(p)}$$

(9)

This transformation allows us to search for short paths connecting $s$ and $d$ in the main network that capture high transition probabilities. We adopt a fast heuristics proposed in [12] to extract these paths. With the subgraph consisting of the extracted paths in the main network, we then apply CROSSQUERY-BASIC on the corresponding NoN to retrieve the top-$k$ most relevant nodes from the target domain-specific network.

Algorithm 3 summarizes this approach. The algorithm starts the shortest path search rooted from $s$ and $d$ simultaneously in $G$ (e.g., using Dijkstra’s algorithm). Let $N(s)$ and $N(t)$ be the neighborhoods of $s$ and $d$ expanded by the algorithm so far. The first overlapping node in $N(s)$ and $N(t)$ identifies the shortest path between $s$ and $d$. Denote this path as $p'_{\text{opt}}$, as it has the maximal transition probability from $s$ to $d$ (Steps 4 to 6). The algorithm continues expanding and adding new nodes in $N(s)$ and $N(t)$. The overlapping nodes in $N(s)$ and $N(t)$ will generate new paths. The expansion stops when the transition probability of the newly discovered path $p'_{\text{opt}}$ drops below $\epsilon \times \text{Prob}(p'_{\text{opt}})$, where $0 < \epsilon < 1$ is an error factor. Thus we have $\text{Prob}(p'_{\text{opt}}) \geq \text{Prob}(p'_{\text{opt}}) - \epsilon$, where $\epsilon$ is the weighed length of a path. Therefore, the neighborhoods with radius $\frac{\epsilon}{\text{Prob}(p'_{\text{opt}})}$ from $s$ and $d$ (i.e., $r_s$ and $r_d$) will contain all significant paths based on $\epsilon$ (Steps 7 to 12). A node $u$ in the neighborhoods can be pruned if $L(s,u) + L(u,d) \leq L(p'_{\text{opt}}) - \log(e)$, where $L(s,u)$ and $L(u,d)$ are the shortest distances between $s$ and $u$ and $u$ and $d$, respectively (Step 14). Finally we apply CROSSQUERY-BASIC on the pruned NoN (Step 15 to 16).

**Time Complexity.** Let the number of nodes and edges in the main network be $g$ and $z$ respectively, and $T$ be the number of iterations. Subgraph extraction takes $O(T(g \log(g) + z))$ using Dijkstra’s algorithm. The pruning step takes $O(g)$. Thus the overall complexity of subgraph extraction is $O(T(g \log(g) + z) + g)$. Note that since $g$ and $z$ are typically very small compared to the overall size of the domain-specific networks, this step is usually very efficient. Experimental results show that CROSSQUERY-Fast brings 4.5 to 7.5x additional speedup while preserving a high accuracy.

**5. EXPERIMENTS**

In this section, we perform comprehensive experiments to evaluate the performance of the proposed methods. We study NoN in two real-world applications: ranking authors in the network of co-author networks constructed from the DBLP bibliographic data, and prioritizing candidate gene in the network of tissue-specific protein interaction networks. Synthetic datasets are also generated to evaluate the efficiency of the developed optimization strategies. All experiments are performed on a 3.00 GHz desktop PC with 8G memory.

**5.1 Co-Author NoN**

We construct the co-author NoN from the DBLP data [24] as follows. The main network consists of 5 research areas, i.e., data mining (DM), machine learning (ML), database (DB), information retrieval (IR), and bioinformatics (BIO). Each area corresponds to a node in the main network. The conferences included in each area are summarized in Table 2. The similarity between two areas is measured by the ratio of the number of citations across these
two areas and the total number of cited papers in them. This is intuitive since two similar areas are more likely to cite each other’s papers. The co-author network in each area is used as the domain-specific network.

5.1.1 Effectiveness Evaluation

To evaluate the effectiveness of the NoN model, we first apply CrossRank and study how the top-10 authors in DB change when varying α. Table 3 shows the ranking results for different α values. Note that when α = 0, the ranking result is the same as that of applying random walk with restart in each individual area independently.

From the table, we can see that the rank of three authors, Jiawei Han, Christos Faloutsos, and Philip S. Yu, increases as α increases. When α = 0, their rank indicates their contributions to DB only. When α increases, their contributions to areas related to DB are also taken into consideration. In particular, they are the top-3 researchers in DM, which is the most similar area to DB in the main network. This indicates that CrossRank can effectively recognize a researcher’s broader impact by incorporating his/her contribution across multiple areas.

Next, we evaluate the effectiveness of CrossQuery by studying an interesting cross-area link prediction task, i.e., for a given query author, we would like to identify future collaborators for this author in a relevant area. In particular, we partition the DBLP dataset into two parts, one from 2001 to 2005 (time interval T1) and another from 2006 to 2010 (time interval T2). We are interested in the DB researchers who have no DM publications during T1 but collaborate with DM researchers and publish in DM during T2. We select such DM-DB author-pairs and use network in T1 to predict their co-authorship in T2.

For comparison, we select methods developed for (1) a single merged co-author network, and (2) heterogeneous information network. For a single merged network, we select the following methods. The path counting (PC) method uses the number of paths between each pair. Here we consider paths with length no longer than 5. The Katz method [11] is also a path counting method that penalizes longer paths. Personalized PageRank [9] and PropFlow [16] are random walk based methods. PathSim is designed for heterogeneous information network [22]. We use the 9 meta paths proposed in [21] for co-authorship link prediction. The parameters of the selected methods are tuned for their optimal performance.

Note that for all the selected methods, we focus on authors with at least certain number of publications, and author-pairs who are within a few hops in the merged overall co-author networks. This is reasonable since highly productive authors are more likely to cooperate with authors within a small distance [21].

We perform leave-one-out cross validation and use accuracy and AUC value as the evaluation criteria. Specifically, we test one selected DM-DB author-pair at each time. The DM researcher is used as the query node and DB is the target query area. A prediction is considered accurate if the test DB researcher is among the top-20 ranked authors in DB.

Table 4: Co-authorship prediction results

<table>
<thead>
<tr>
<th>Hops</th>
<th>Methods</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3, 4]</td>
<td>PC</td>
<td>0.6521</td>
</tr>
<tr>
<td>Katz</td>
<td>0.643</td>
<td>0.3714</td>
</tr>
<tr>
<td>PropFlow</td>
<td>0.6850</td>
<td>0.3478</td>
</tr>
<tr>
<td>PathSim</td>
<td>0.6743</td>
<td>0.3478</td>
</tr>
<tr>
<td>PageRank</td>
<td>0.7099</td>
<td>0.3478</td>
</tr>
<tr>
<td>CrossQuery</td>
<td>0.6308</td>
<td>0.2895</td>
</tr>
<tr>
<td>[3, 6]</td>
<td>PC</td>
<td>0.7409</td>
</tr>
<tr>
<td>Katz</td>
<td>0.5786</td>
<td>0.2536</td>
</tr>
<tr>
<td>PropFlow</td>
<td>0.5950</td>
<td>0.2895</td>
</tr>
<tr>
<td>PathSim</td>
<td>0.4362</td>
<td>0.1053</td>
</tr>
<tr>
<td>PageRank</td>
<td>0.5880</td>
<td>0.2368</td>
</tr>
<tr>
<td>CrossQuery</td>
<td>0.6308</td>
<td>0.2895</td>
</tr>
</tbody>
</table>

Table 4 shows the results. From the table, we can see that CrossQuery outperforms all alternative methods. The performance gain becomes larger when more test author-pairs are available. Alternative methods are not suitable for this cross-area query task, since they only consider the similarity between the authors but not areas. Note that the meta paths used in PathSim do not consider the similarity between areas and conferences either.

5.1.2 Efficiency Evaluation

We evaluate the efficiency of the proposed methods using both the DBLP and synthetic datasets. The main network in the DBLP NoN consists of 121 conferences. The number of nodes in the conference-specific coauthor networks ranges from 88 to 14,636 depending on the size of the conferences, with a total of 259,822 nodes. The synthetic NoN are generated using the R-MAT model [2] so that the resulting networks resemble real-world networks. The main network consists of 1,023 nodes. The domain-specific networks contain 935 to 8,100 nodes with a total of 3,773,519 nodes.

Figure 2 shows the running time of CrossQuery-Basic and CrossQuery-Fast when varying the number of returned nodes k. The running time of CrossRank is also reported in the figure as a reference. The running time is averaged over 10 randomly selected query nodes and target domain-specific networks. The parameters are a = 0.2, c = 0.85, and ϵ = 10^-3. It can be seen that both CrossQuery-Basic and CrossQuery-Fast are much more efficient than CrossRank. The running time of CrossQuery-Basic and CrossQuery-Fast increases as k increases. The randomly selected query nodes may affect the trend, e.g., in Figure 2(b) when k = 500. The corresponding averaged accuracy of CrossQuery-Fast is shown in 2(c). From the results, we can see that CrossQuery-Fast achieves more than 90% accuracy for all settings and dramatically improves the efficiency.

5.2 Protein Interaction NoN

In this section, we apply the proposed NoN model to the candidate gene prioritization problem, which has recently attracted in-
associations between 106 diseases and 102 genes. These genes are
considered as a successful prediction. This evaluation method is used as seed nodes in CROSSRANK which are represented by the white rectangles in Figure 3(b).

We select four state-of-the-art methods for comparison, including RWRH [14], BIRW [31], PRINCE [28] and Katz [20]. We use the standard leave-one-out cross validation to compare the prioritization accuracy of the selected methods. Specifically, at each time, we remove an association between the query disease and one causal gene, as well as all the associations involving the causal gene to avoid the trivial cases where mutations in the same gene cause two very similar diseases [28, 17]. This causal gene is used as the test gene. If the test gene is ranked within the top-\(k\) genes, where \(k\) is the total number of known genes for the query disease, this is considered as a successful prediction. This evaluation method is
We further study how often CROSSRANK gives higher ranks to test genes than other methods. The results are shown in Table 6. The Wilcoxon signed rank test is used to assess the statistical significance of the difference between the ranking lists given by the two compared methods. It is clear from the table that the results of CROSSRANK are significantly better than that of the alternatives.

6. RELATED WORK

Networks are ubiquitous in real-life applications. The simplest model uses a single graph to represent a network. A variety of ranking algorithms have been developed for a single network [9, 8, 19, 6, 15]. Recently, various advanced network models have been proposed, such as multi-relational network [18], heterogeneous information network [23], and hypergraph [34]. The multi-relational network and heterogeneous information network can incorporate node or edge type information: in a multi-relational network, edges connecting two nodes may be of different types; while different types of objects can co-exist in a heterogeneous information network. Some work uses them interchangeably [32]. A tensor based co-ranking framework is proposed in [18] for multi-relational network. In heterogeneous information network, ranking-based clustering [23] and ranking-based classification [10] frameworks are developed, where ranking and clustering/classification can be mutually enhanced. In a hypergraph, an edge can connect any subset of nodes and is similar to a main node in our NoN model. But the set of nodes connected by an edge in a hypergraph do not form any network topology. If we treat each hyperedge as a main node, hypergraph can be viewed as special case of NoN where we do not have any links among domain nodes at all.

7. CONCLUSION

Ranking is a primitive operation in network analysis. In this paper, we propose a new network data model, Network of Networks (NoN), which enables novel ranking tasks such as CROSSRANK and CROSSQUERY. We formulate ranking on NoN as a regularized optimization problem, develop efficient algorithms, and provide rigorous theoretical analysis. Experimental results on real-world datasets demonstrate the effectiveness and efficiency of the proposed methods.

8. ACKNOWLEDGEMENT

This work was partially supported by the National Science Foundation grants IIS-1162374, IIS-1218036 and IIS1017415, by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, by Defense Advanced Research Projects Agency (DARPA) under Contract Number W911NF-11-C-0200 and
9. REFERENCES