FIRST: Fast Interactive Attributed Subgraph Matching

Boxin Du
Arizona State University
boxin.du@asu.edu

Si Zhang
Arizona State University
szhan172@asu.edu

Nan Cao
Tongji University
nan.cao@gmail.com

Hanghang Tong*
Arizona State University
hanghang.tong@asu.edu

ABSTRACT

Attributed subgraph matching is a powerful tool for explorative mining of large attributed networks. In many applications (e.g., network science of teams, intelligence analysis, finance informatics), the user might not know what exactly s/he is looking for, and thus require the user to constantly revise the initial query graph based on what s/he finds from the current matching results. A major bottleneck in such an interactive matching scenario is the efficiency, as simply rerunning the matching algorithm on the revised query graph is computationally prohibitive. In this paper, we propose a family of effective and efficient algorithms (FIRST) to support interactive attributed subgraph matching. There are two key ideas behind the proposed methods. The first is to recast the attributed subgraph matching problem as a cross-network node similarity problem, whose major computation lies in solving a Sylvester equation for the query graph and the underlying data graph. The second key idea is to explore the smoothness between the initial and revised queries, which allows us to solve the new/updated Sylvester equation incrementally, without re-solving it from scratch. Experimental results show that our method can achieve (1) up to 16× speed-up when applying on networks with 6M+ nodes; (2) preserving more than 90% accuracy compared with existing methods; and (3) scales linearly with respect to the size of the data graph.

KEYWORDS

Interactive attributed subgraph matching; cross-network similarity; Inexact matching

1 INTRODUCTION

Many real networks often accompany with rich node and/or edge attribute, including the demographic information for users on a social network (i.e., node attribute), the transaction types on a financial transaction network (i.e., edge attribute), the expertise of team members as well as the communication channels between them on a collaboration network (i.e., both node and edge attributes). Attribute subgraph matching [21, 27] is the key for many explorative mining tasks, i.e., to help identify *user-specific* patterns from such attributed networks, and has become an integral part of some emergent visual graph analytic platforms [6]. To name a few, in network science of teams, attributed subgraph matching is the cornerstone to help form a team of experts with desired skills of each member as well as the communication pattern between team members (i.e., example-based team formation) [15, 16]; in finance informatics, it is a powerful tool to identify suspicious transaction patterns (e.g., money laundry ring) [27]; in intelligence analysis and law enforcement, it can help end-analyst generate valuable leads (e.g., a suspicious terror plot, the master-criminal mind, etc.)[26].

Despite that tremendous progress has been made (See Section 5 for a review), most, if not all, of the existing attributed subgraph matching algorithms requires the user accurately knows what s/he is looking for, in other words, to provide an accurate query graph. However, in some application scenarios, the end-user might only have a vague idea on her search intent at the beginning and thus needs to constantly revise and refine her initial query graph. Figure 1 represents an illustrative example of such interactive matching process.

On the left side of Figure 1 is the input data (attributed) data network[1], and on the right side we illustrate a procedure of team formation in an interactive style. Let us assume skill A to skill D stands for *programming*, *databases*, *machine learning* and *visualization*, respectively, and the edge attribute 1 to 3 represents three different communication approaches. We have the following interactive matching process. (1) At the beginning, the user only knows s/he wants to form a team of size 3, with two skills (e.g., programming and machine learning) and the team should be led by the machine learning expert, and therefore s/he issues a line query (Q_1). (2) After the user sees the initial matching graph (M_1), s/he realizes that the project only needs one programmer; but in the meanwhile it requires another expert in *databases* and better communication between all the team members. Therefore, s/he issues a revised clique query (Q_2). (3) After seeing the corresponding updated matching subgraph (M_2), s/he decides to expand the team size by adding an additional expert in data visualization to help databases expert. Thus, s/he expands the previous query graph by adding an additional link and node (Q_3). (4) After s/he sees the updated matching result (M_3), s/he finds that having too many communications (i.e., over-communications) between the team members might hurt the team productivity. Thus, s/he revises the query graph again (Q_4), to keep only vital communications

*To whom correspondence should be addressed

[1]In this paper, we use the terms 'graph' and 'network' interchangeably.

KDD '17, August 13-17, 2017, Halifax, NS, Canada
© 2017 ACM. 978-1-4503-4887-4/17/08…$15.00
DOI: 10.1145/3097983.3098040

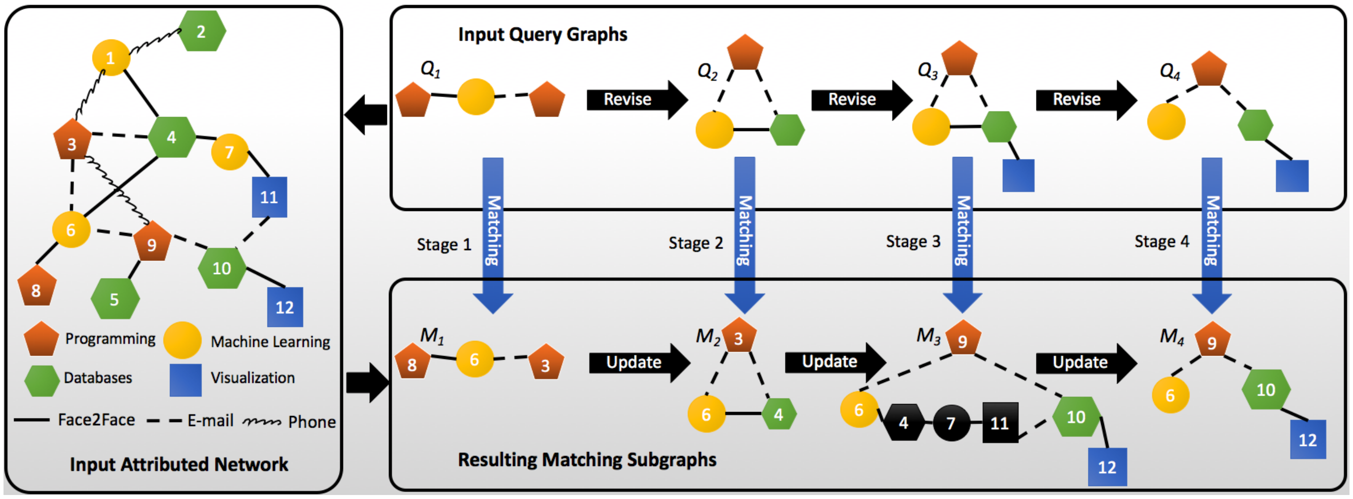


Figure 1: An illustrative example of interactive attributed subgraph matching. Best viewed in color.

between the key team members. Finally the user finds the ideal team (M_4).

In such an interactive setting, a major bottleneck is the computational efficiency. This is because simply re-running the matching algorithms on the revised query graph from scratch might be computationally too costly as such algorithms require either building an index of the underlying data graph (e.g., [25]) or a costly iterative process during the query stage (e.g., [21, 27]).

To address these limitations, we propose a family of effective and efficient algorithms (FIRST) to support interactive attributed subgraph matching scenario. There are two key ideas behind the proposed methods. The first is to recast the attributed subgraph matching problem as a cross-network node similarity problem. This formulation allows us to simultaneously encode topology consistency and attribute consistency in a coherent optimization problem, whose major computation lies in solving a Sylvester equation for the query and the underlying data graph [29]. The second key idea is to explore the smoothness between the initial and revised queries, which means that the revised query can often be viewed as a perturbed version of the previous query graph. For example in Figure 1, the third query graph (Q_3) can be viewed as perturbed version of the second query (Q_2) with an additional node with one additional link. It turns out this observation enable us to solve the new/updated Sylvester equation incrementally, without re-solving it from scratch. The proposed FIRST algorithms enjoy a *linear* time complexity with respect to the input data network size. We conduct extensive experiments on real-world datasets, which show that the proposed method leads to up to 16× speed-up with more than 90% accuracy.

The rest of paper is organized as follows. Section 2 formulates the problem of interactive attributed subgraph matching. Section 3 presents our proposed FIRST algorithms. Section 4 presents the experimental results on real-world datasets. We review the related work in Section 5 and conclude the paper in section 6.

2 PROBLEM DEFINITION

Table 1 summarizes the main symbols and notation used throughout the paper. We use bold uppercase letters for matrices (e.g., $\mathbf{A}$), bold lowercase letters for vectors (e.g., $\mathbf{s}$), and lowercase letters (e.g., α) for scalars. We use the calligraphic letter $\mathcal{G}$ to represent an attributed network, i.e., $\mathcal{G} = (\mathbf{A}, \mathbf{N}_\mathbf{A}, \mathbf{E}_\mathbf{A})$, where $\mathbf{A}$ is the adjacency matrix, $\mathbf{N}_\mathbf{A}$ and $\mathbf{E}_\mathbf{A}$ are the node and edge attribute matrices of $\mathcal{G}$, respectively. We use the subscript q to denote the corresponding notations for the query graph (i.e., $\mathcal{Q} = (\mathbf{A}_q, \mathbf{N}_q, \mathbf{E}_q)$), and $\tilde{}$ to denote the corresponding notation after the user modifies the initial query (i.e., $\tilde{\mathcal{Q}} = (\tilde{\mathbf{A}}_q, \tilde{\mathbf{N}}_q, \tilde{\mathbf{E}}_q)$ is the revised query graph). Likewise, the initial and updated similarity matrix are denoted by $\mathbf{s}$ and $\tilde{\mathbf{s}}$. The initial matching subgraph and the updated matching subgraph are denoted by $\mathcal{M}$ and $\tilde{\mathcal{M}}$ respectively. Additionally, We use $\hat{}$ to denote the approximate version of vectors or matrices in this paper (e.g., $\hat{\mathbf{s}}$, $\hat{\mathbf{W}}^{sym}$, etc.).

The node attribute matrix of input networks $\mathbf{N}$ is defined as $\mathbf{N} = \sum_{p=1}^{K} \mathbf{N}_\mathbf{A}^p \otimes \mathbf{N}_\mathbf{q}^p$ where K is the number of distinct node labels. $\mathbf{N}_\mathbf{A}^p$ and $\mathbf{N}_\mathbf{q}^p$ are diagonal matrices in which $\mathbf{N}_\mathbf{A}^p(a,a) = 1$ if the node a in network $\mathcal{G}$ has node attribute k and otherwise it is equal to 0. The edge attribute matrix of input networks $\mathbf{E}$ is defined as $\mathbf{E} = \sum_{l=1}^{L} \mathbf{E}_\mathbf{A}^l \otimes \mathbf{E}_\mathbf{q}^l$ where L is the number of distinct edge labels. $\mathbf{E}_\mathbf{A}^l$ and $\mathbf{E}_\mathbf{q}^l$ are $n \times n$ and $k \times k$ matrices respectively. $\mathbf{E}_\mathbf{A}^l(a,b) = 1$ if the edge (a,b) in network $\mathcal{G}$ has edge attribute l and otherwise it is equal to 0. For example, for the initial query graph ($\mathcal{Q}_1$) in Fig. 1, we have $\mathbf{N}_\mathbf{q}^1(1,1) = 1$, $\mathbf{N}_\mathbf{q}^2(1,1) = 0$, $\mathbf{E}_\mathbf{q}^1(1,2) = 1$ and $\mathbf{E}_\mathbf{q}^2(1,2) = 0$, etc. For the revised query graph, we have $\mathbf{E}_\mathbf{q}^1(2,3) = 1$ and $\mathbf{N}_\mathbf{q}^3(3,3) = 1$, etc.

With these notations, the interactive attributed subgraph matching problem can be formally defined as:

Problem 1. *Interactive Attribute Subgraph Matching.* ***Given:*** *(1) an undirected attributed network $\mathcal{G}$, (2) an undirected initial query graph $\mathcal{Q}$, (3) the initial matching subgraph $\mathcal{M}$, (4) the*

revised query graph $\tilde{\mathcal{Q}}$;
Output: *the updated matching subgraph $\tilde{\mathcal{M}}$.*

For the team formation example in Figure 1, between stage-1 and stage-2, the line query (Q_1) and the clique query (Q_2) are the initial and the revised query graphs, respectively; and M_1 and M_2 are the corresponding initial and revised matching subgraph, respectively. Between stage-2 and stage-3, the clique query (Q_2) will be treated as the initial query graph, and Q_3 becomes the new revised query graph, so on and so forth.

Table 1: Symbols and Definition

Symbols	Definition
$\mathcal{G} = \{\mathrm{A}, \mathrm{N}, \mathrm{E}\}$	an attributed network
$\mathcal{Q}, \tilde{\mathcal{Q}}$	initial and revised query network
$\mathcal{M}, \tilde{\mathcal{M}}$	initial and updated matching subgraph
$\mathrm{A}, \mathrm{A_q}$	the adjacency matrix of the attributed data network and query graph
$\mathrm{N_A}/\mathrm{N_q}, \mathrm{E_A}/\mathrm{E_q}$	the node and edge attribute matrix of the network/query graph
n, k	# of nodes in $\mathcal{G}$ and $\mathcal{Q}$
m_1, m_2	# of edges in $\mathcal{G}$ and $\mathcal{Q}$
P, L	# of the node and edge labels
a, b	node/edge indices of $\mathcal{G}$
x, y	node/edge indices of $\mathcal{Q}$
p, l	node/edge label indices
$\mathbf{I}$	an identity matrix
$\mathbf{H}$	$k \times n$ prior alignment preference
S	$k \times n$ similarity matrix
r, t	reduced ranks
α	the parameter, $0 < \alpha < 1$
$\mathrm{s} = \mathrm{vec}(\mathrm{S})$	vectorize a matrix S in column order
$\mathrm{Q} = \mathrm{mat}(\mathbf{q}, n_2, n_1)$	reshape vector $\mathbf{q}$ into an $n_2 \times n_1$ matrix in column order
$\mathbf{W}^{sym}$	symmetrically normalize matrix $\mathbf{W}$
$\mathrm{D} = \mathrm{diag}(\mathbf{d})$	diagonalize a vector $\mathbf{d}$
$\otimes$	Kronecker product
$\odot$	element-wise matrix product
abs()	absolute value
$\lVert \cdot \rVert_F$	Frobenius norm
and()	AND operation

3 FAST INTERACTIVE ALGORITHMS

In this section, we first review an existing network alignment algorithm, which provides the base for our proposed algorithm. Then, we present our algorithms (FIRST) in different scenarios, e.g., revising the topology/node attributes/edge attributes in the query graph, whether the input data network has both node and edge attributes, etc.

3.1 Preliminaries

Generally speaking, in attributed subgraph matching, we want to find a subgraph from the input data network $\mathcal{G}$ that maximizes some "goodness" function with regard to the query graph $\mathcal{Q}$ [27] . Here, our idea is to recast it as a cross-network node similarity problem. To be specific, let S be a $k \times n$ non-negative cross-network node-similarity matrix, where $\mathrm{S}(i, j)$ measures the cross-network similarity between the i^{th} query node in $\mathcal{Q}$ and the j^{th} node in $\mathcal{G}$ (i.e., to what extent the j^{th} node in $\mathcal{G}$ matches the i^{th} query node).

In order to find the cross-network node similarity matrix S, we adopt a recent network alignment algorithm [29], which naturally encodes both the topological and attribute consistency between two networks (the data network and the query graph in our setting) in the following Sylvester equation (please refer to [29] for the full details).

$$\mathbf{s} = \alpha \mathbf{W}^{sym}\mathbf{s} + (1 - \alpha)\mathbf{h} \tag{1}$$

where $\mathbf{s} = \mathrm{vec}(\mathrm{S})$, $\mathbf{h} = \mathrm{vec}(\mathbf{H})$ and $\mathbf{H}$ is a $k \times n$ matrix of prior similarity knowledge. $\mathbf{W}^{sym} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ is the symmetrical normalization of $\mathbf{W}$ and $\mathbf{W}$ can take three possible forms according to the availability of the attribute information in the networks [29], including

(i) $\mathbf{W}^{sym} = \mathrm{A} \otimes \mathrm{A_q}$: if only adjacency matrix is available;
(ii) $\mathbf{W}^{sym} = \mathrm{N}(\mathrm{A} \otimes \mathrm{A_q})\mathrm{N}$: if the adjacency matrix and node attributes are available but edge attributes are missing;
(iii) $\mathbf{W}^{sym} = \mathrm{N}[\mathbf{E} \odot (\mathrm{A} \otimes \mathrm{A_q})]\mathrm{N}$: if the adjacency matrix, node and edge attributes are all available.

And $\mathbf{D}$ is the diagonal degree matrix of $\mathbf{W}$. For example, if both node and edge attributes are available ($\mathbf{W}^{sym}$ being type (iii)), $\mathbf{D}$ is computed by:

$$\mathbf{D} = \mathrm{diag}(\sum_{k,k'=1}^{K}\sum_{l=1}^{L}(\mathbf{N}_A^k(\mathbf{E}_A^l \odot \mathbf{A})\mathbf{N}_A^{k'}\mathbf{1}) \otimes (\mathbf{N}_q^k(\mathbf{E}_q^l \odot \mathbf{A}_q)\mathbf{N}_q^{k'}\mathbf{1})) \tag{2}$$

where K and L are the number of different node and edge labels respectively.

The solution of Eq. (1) can be obtained by either an iterative procedure or a closed-form formula. And the closed-form solution can be further approximated (and sped up) by using low-rank approximation on the two input networks. However, none of these solutions is applicable in the interactive setting. This is because: (1) for the iterative solution, each iteration requires $O(\min(km_1, nm_2)$ time complexity and it might take many iterations to compute the similarity matrix S only for the initial query network, let alone for the interactively updated query networks, and (2) for the closed-form solution, its $O(n^3k^3)$ time complexity, or even its approximate solution, is still computationally too costly if user frequently changes the queries.

The common key ideas behind our upcoming proposed algorithm FIRST are that: (1) we recast the attributed subgraph matching problem as a cross-network node similarity problem (i.e., to compute the matrix S in Eq. (1)), and (2) by exploring the smoothness between the initial query and updated queries, we can solve the Sylvester equation incrementally. In our paper, we assume that the data network $\mathcal{G}$ and the prior preference $\mathbf{H}$ remain unchanged during the interactive process. In practice, the size of the query network is often much smaller than that of the data network, i.e. $k \ll n$.

3.2 Handling Node Attribute

In this subsection we consider the scenario in which node attribute is available but edge attribute is missing in both network and query graph. We discuss two cases: (A) only revising the topology of the query graph and (B) only revising node attribute of the query graph. First we present Algorithm 1 to solve the case where only topology is changed.

A - Topology Change. Based on our problem formulation and assumptions, in the interactive scenario the updated similarity vector $\tilde{\mathrm{s}}$ after query modification can be expressed as follows:

$$\tilde{\mathrm{s}} = (1 - \alpha)(\mathbf{I} - \alpha\tilde{\mathbf{W}}^{sym})^{-1}\mathbf{h} \tag{3}$$

where $\tilde{\mathbf{W}}^{sym} = \mathrm{D}^{-1/2}\mathrm{N}(\mathrm{A} \otimes \mathrm{A_q})\mathrm{N}\mathrm{D}^{-1/2}$. Since only the topology of the updated query differs from initial query, the node attribute information N will not contribute to $\tilde{\mathrm{s}}$.

Since many real-world networks are observed to have a low-rank structure, we can leverage this characteristics to obtain a good approximation of the similarity matrix. Here, we define the approximate similarity matrix as follows:

Definition 1. *(APPROXIMATE SIMILARITY VECTOR) Given a similarity vector* $\mathbf{s} = (1-\alpha)(\mathbf{I}-\alpha\mathbf{W}^{sym})^{-1}\mathbf{h}$*, its approximate similarity vector* $\hat{\mathbf{s}}$ *is given by*

$$\hat{\mathbf{s}} = (1-\alpha)(\mathbf{I}-\alpha\hat{\mathbf{W}}^{sym})^{-1}\mathbf{h}$$

where $\hat{\mathbf{W}}^{sym}$ *is a low rank approximation of* $\mathbf{W}^{sym}$*.*

Since the adjacency matrices $\mathbf{A}$ and $\mathbf{A_q}$ are both symmetric, we can apply rank-r eigenvalue decomposition (EVD) on $\mathbf{A}$ and $\mathbf{A_q}$. An additional advantage of using EVD is that we can reduce the space complexity by only storing the low-rank matrices instead of the whole adjacency matrices. The algorithm is summarized in Algorithm 1.

Algorithm 1 FIRST-Q

```
Input: the attributed data network $\mathcal{G} = \{\mathbf{A}, \mathbf{N_A}\}$,
1: the initial and revised query network $\mathcal{Q} = \{\mathbf{A_q}, \mathbf{N_q}\}$, $\tilde{\mathcal{Q}} = \{\tilde{\mathbf{A}}_\mathbf{q}, \tilde{\mathbf{N}}_\mathbf{q}\}$,
2: the alignment preference matrix $\mathbf{H}$,
3: parameter $\alpha$.
Output: Approx. updated similarity matrix $\tilde{\mathbf{S}}$.
4: Precomputing Stage:
5: $\mathbf{U_A\Lambda_A U_A^T} \leftarrow \mathbf{A}$; //top r eigenvalue decomposition
6: $\mathbf{U_Q\Lambda_Q U_Q^T} \leftarrow \mathbf{A}_q$; //top t eigenvalue decomposition
7: Store $\mathbf{U_A}, \mathbf{\Lambda_A}$;
8: Interactive Stage:
9: $\mathbf{U_Q\Lambda_Q U_Q^T} \leftarrow \tilde{\mathbf{A}}_\mathbf{q}$; //top t eigenvalue decomposition
10: Compute node attribute matrix of input networks $\mathbf{N}$ and diagonal degree
    matrix $\mathbf{D}$; Construct $\mathbf{P} = \mathbf{D}^{-\frac{1}{2}}\mathbf{N}$, $\mathbf{D_1} = \mathbf{P}^{-1}\mathbf{P}^{-1}$;
11: $\mathbf{L} \leftarrow \mathbf{U_A} \otimes \mathbf{U_Q}$;
12: $\mathbf{R} \leftarrow \mathbf{U_A^T} \otimes \mathbf{U_Q^T}$;
13: $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda_A} \otimes \mathbf{\Lambda_Q}$;
14: $\tilde{\mathbf{s}} = (1-\alpha)\mathbf{P}^{-1}[\mathbf{D_1}^{-1} + \alpha\mathbf{D_1}^{-1}\mathbf{L}(\mathbf{\Lambda}^{-1} - \alpha\mathbf{R}\mathbf{D_1}^{-1}\mathbf{L})^{-1}\mathbf{R}\mathbf{D_1}^{-1}]\mathbf{P}^{-1}\mathbf{h}$;
15: $\tilde{\mathbf{S}} = mat(\tilde{\mathbf{s}}, n, k)$; //reshape similarity vector
```

From the fourth line to seventh line are the precomputing stage. The top r eigenvalue decomposition of $\mathbf{A}$ and the top t eigenvalue decomposition of $\mathbf{A}_q$ are calculated. $\mathbf{U_A}$ and $\mathbf{\Lambda_A}$ are stored. In the interactive stage, only the top t eigenvalue decomposition of $\tilde{\mathbf{A}}_q$ is calculated. Then $\tilde{\mathbf{S}}$ is computed from line 10 to line 15. The proof of correctness of FIRST-Q is presented as follows:

Theorem 3.1 (Correctness of FIRST-Q). *The Algorithm 1 (FIRST-Q) gives the approximate similarity vector by Definition 1:* $\tilde{\mathbf{s}} = \hat{\mathbf{s}}$.

Proof According to Definition 1,

$$\begin{aligned}\hat{\mathbf{s}} &= (1-\alpha)[\mathbf{I} - \alpha\mathbf{P_1}(\hat{\mathbf{A}} \otimes \hat{\mathbf{A}}_\mathbf{q})\mathbf{P_1}]^{-1}\mathbf{h}\\ &= (1-\alpha)\mathbf{P_1}^{-1}[\mathbf{D_1} - \alpha(\hat{\mathbf{A}} \otimes \hat{\mathbf{A}}_\mathbf{q})]^{-1}\mathbf{P_1}^{-1}\mathbf{h}\end{aligned} \tag{4}$$

in which $\mathbf{P_1} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{N} = \mathbf{N}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, $\mathbf{D_1} = \mathbf{P_1}^{-1}\mathbf{P_1}^{-1}$. Let $\mathbf{U_A\Lambda_A U_A^T}$ be top r eigenvalue decomposition of $\mathbf{A}$ and $\mathbf{U_Q\Lambda_Q U_Q^T}$ be top t eigenvalue decomposition of $\mathbf{A}_q$. Then in the interactive stage, $\hat{\mathbf{W}}^{sym}$ can be written as:

$$\begin{aligned}\hat{\mathbf{W}}^{sym} &= \tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{N}[(\mathbf{U_A\Lambda_A U_A^T}) \otimes (\mathbf{U_Q\Lambda_Q U_Q^T})]\mathbf{N}\tilde{\mathbf{D}}^{-\frac{1}{2}}\\ &= \tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{N}[(\mathbf{U_A} \otimes \mathbf{U_Q})(\mathbf{\Lambda_A} \otimes \mathbf{\Lambda_Q})(\mathbf{U_A^T} \otimes \mathbf{U_Q^T})]\mathbf{N}\tilde{\mathbf{D}}^{-\frac{1}{2}}\end{aligned} \tag{5}$$

Let $\mathbf{L} = \mathbf{U_A} \otimes \mathbf{U_Q}$, $\mathbf{R} = \mathbf{U_A^T} \otimes \mathbf{U_Q^T}$, $\mathbf{\Lambda} = \mathbf{\Lambda_A} \otimes \mathbf{\Lambda_Q}$. According to equation 4 and the definition of $\hat{\mathbf{s}}$,

$$\begin{aligned}\hat{\mathbf{s}} &= (1-\alpha)(\mathbf{I} - \alpha\mathbf{P_1}\mathbf{L\Lambda R}\mathbf{P_1})^{-1}\mathbf{h}\\ &= (1-\alpha)\mathbf{P_1}^{-1}(\mathbf{D_1} - \alpha\mathbf{L\Lambda R})^{-1}\mathbf{P_1}^{-1}\mathbf{h}\\ &= (1-\alpha)\mathbf{P_1}^{-1}[\mathbf{D_1}^{-1} + \alpha\mathbf{D_1}^{-1}\mathbf{L}(\mathbf{\Lambda}^{-1} - \alpha\mathbf{R}\mathbf{D_1}^{-1}\mathbf{L})^{-1}\mathbf{R}\mathbf{D_1}^{-1}]\mathbf{P_1}^{-1}\mathbf{h}\end{aligned} \tag{6}$$

where the third equality comes from the Sherman-Morrison Lemma [20]. Hence the correctness of FIRST-Q is proved. □

It is worth pointing out that if the node attribute information is also missing, which means that $\mathbf{W}^{sym}$ takes the form of type (i), the algorithm also works by setting $\mathbf{N}$ to be identity matrix $\mathbf{I}$. The proof is almost identical to the proof above.

B - Node Attribute Change. Here, we provide an algorithm (FIRST-N) for the scenario where only node attribute of the query graph is revised during user's interactive query process. Again, the edge attribute is not available in this scenario (i.e., no $\mathbf{E}$ and $\mathbf{E_q}$). The algorithm is summarized in Algorithm 2.

Algorithm 2 FIRST-N

```
Input: the attributed data network at time step 1 $\mathcal{G} = \{\mathbf{A}, \mathbf{N_A}\}$,
1: the initial and revised query network $\mathcal{Q} = \{\mathbf{A_q}, \mathbf{N_q}\}$, $\tilde{\mathcal{Q}} = \{\tilde{\mathbf{A}}_\mathbf{q}, \tilde{\mathbf{N}}_\mathbf{q}\}$,
2: the alignment preference matrix $\mathbf{H}$,
3: parameter $\alpha$.
Output: Approx. updated similarity matrix $\tilde{\mathbf{S}}$.
4: Precomputing Stage:
5: $\mathbf{U_A\Lambda_A U_A^T} \leftarrow \mathbf{A}$; //top r eigenvalue decomposition;
6: $\mathbf{U_Q\Lambda_Q U_Q^T} \leftarrow \mathbf{A}_q$; //top t eigenvalue decomposition;
7: $\mathbf{L} \leftarrow \mathbf{U_A} \otimes \mathbf{U_Q}$;
8: $\mathbf{R} \leftarrow \mathbf{U_A^T} \otimes \mathbf{U_Q^T}$;
9: $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda_A} \otimes \mathbf{\Lambda_Q}$;
10: Store $\mathbf{L}, \mathbf{R}, \mathbf{\Lambda}$;
11: Interactive Stage:
12: Compute node attribute matrix of input matrix $\mathbf{N}$ and diagonal degree
    matrix $\mathbf{D}$ with $\tilde{\mathbf{N}}_q$, $\mathbf{A}$ and $\tilde{\mathbf{A}}_q$; Compute $\mathbf{P} = \mathbf{D}^{-\frac{1}{2}}\tilde{\mathbf{N}}_q$;
13: Compute $\mathbf{D_1} = \mathbf{P}^{-1}\mathbf{P}^{-1}$;
14: $\tilde{\mathbf{s}} = (1-\alpha)\mathbf{P}^{-1}[\mathbf{D_1}^{-1} + \alpha\mathbf{D_1}^{-1}\mathbf{L}(\mathbf{\Lambda}^{-1} - \alpha\mathbf{R}\mathbf{D_1}^{-1}\mathbf{L})^{-1}\mathbf{R}\mathbf{D_1}^{-1}]\mathbf{P}^{-1}\mathbf{h}$;
15: $\tilde{\mathbf{S}} = mat(\tilde{\mathbf{s}}, n, k)$; //reshape similarity vector
```

In the precomputing stage, the algorithm calculates and stores $\mathbf{L}$, $\mathbf{R}$ and $\mathbf{\Lambda}$, while in the interactive stage the algorithm can directly construct $\mathbf{P}$. $\tilde{\mathbf{S}}$ is calculated from line 12 to line 15.

From the algorithm, we can notice that the eigenvalue decomposition of adjacency matrix $\mathbf{A}$ and $\mathbf{A}_q$, the construction of matrices $\mathbf{L}$, $\mathbf{R}$ and $\mathbf{\Lambda}$ can be precomputed at initial step, thus further speed up this algorithm, compared with Algorithm 1. The proof of the correctness of Algorithm 2 is similar to the proof of Algorithm 1 and is omitted for space.

3.3 Handling Edge Attribute

In this subsection, we discuss the interactive scenario where both node and edge attribute are available. Note that in this case the query can be revised in multiple ways (e.g., only revising network topology/node attribute/edge attribute vs simultaneously revising network topology as well as attributes, etc.). In our proposed algorithm (FIRST-E), we consider the general case where network topology, node and edge attributes are all revised simultaneously during one interactive stage. The rest of the ways to revise the query graph are special cases, and thus can also be supported by our approach. The algorithm is presented in Algorithm 3.

Algorithm 3 FIRST-E

```
Input: the attributed network at time step 1 $\mathcal{G} = \{A, N_A, E_A\}$,
 1: the initial and revised query network $\mathcal{Q} = \{A_q, N_q, E_q\}$, $\tilde{\mathcal{Q}} = \{\tilde{A}_q, \tilde{N}_q, \tilde{E}_q\}$
 2: the alignment preference matrix $H$,
 3: parameter $\alpha$, index of changed edge attribute $l'$ (optional).
Output: Approx. updated similarity matrix $\tilde{S}$.
 4: Precomputing Stage:
 5: for each $l \in [1, L]$ do
 6:     $U^l_A \Lambda^l_A (U^l_A)^T \leftarrow E^l_A \odot A$; //top r eigenvalue decomposition
 7:     Store $U^l_A$, $\Lambda^l_A$;
 8: end for
 9: if $l'$ is not empty then
10:     for each $k \in [1, L]$ do
11:         $U^k_q \Lambda^k_q (U^k_q)^T \leftarrow E^k_q \odot A_q$; //top t eigenvalue decomposition
12:         Store $U^k_q$, $\Lambda^k_q$;
13:     end for
14: end if
15: Interactive Stage:
16: Construct $N$ and $D$ from $A$, $\tilde{A}_q$, $N_A$, $\tilde{N}_q$, $E_A$, $\tilde{E}_q$;
17: for each $l \in [1, L]$ (or each $l \in l'$ if $l'$ is not empty) do
18:     $U^l_q \Lambda^l_q (U^l_q)^T \leftarrow E^l_q \odot \tilde{A}_q$; //top t eigenvalue decomposition
19: end for
20: Construct block matrix $U = [V_1, V_2, \ldots, V_L]$, in which $V_i = U^i_A \otimes U^i_q$ ($i \in [1, L]$), and block matrix $\Lambda = diag(Y_1, Y_2, \ldots Y_L)$, in which $Y_j = \Lambda^j_A \otimes \Lambda^j_q$ ($j \in [1, L]$);
21: $L \leftarrow D^{-\frac{1}{2}} NU$;
22: $R \leftarrow U^T ND^{-\frac{1}{2}}$;
23: $\tilde{s} = (1 - \alpha)[I + \alpha L(\Lambda^{-1} - \alpha RL)^{-1}R]h$;
24: $\tilde{S} = mat(\hat{s}, n, k)$; //reshape similarity vector
```

The precomputing stage is from line 4 to line 14 and the interactive stage is from line 16 to line 24. In line 20, two block matrices $(\mathbf{U}, \mathbf{\Lambda})$ are constructed. $\mathbf{U}$ is a $1 \times L$ block matrix with each element being $\mathbf{V_i}$, while $\mathbf{\Lambda}$ is a $L \times L$ diagonal block matrix with each diagonal element being $\mathbf{Y_j}$ ($i, j \in [1, L]$).

As mentioned at the beginning of this subsection, the algorithm still works in other ways to revise the query graph by setting $\tilde{A_q}$, $\tilde{N_q}$ or $\tilde{E_q}$ equal to their initial counterparts. Specifically, if only certain edge attributes are changed, the algorithm could take an optional parameter $\mathbf{l}'$ (line 3) as the index of changed edge attribute, otherwise $\mathbf{l}'$ is set empty. In line 9, the top t eigenvalue decomposition of the element-wise product of the k^{th} edge attribute matrix $\mathbf{E^k_q}$ and network adjacency matrix $\mathbf{A}$ is computed, if $\mathbf{l}'$ is not empty. In the following interactive stage, the eigenvalue decomposition can be only calculated on the element-wise product of the changed edge attribute matrices $(\mathbf{E^l_q})$ and $\tilde{\mathbf{A}}_\mathbf{q}$ (line 17 to 19), which further speed up the interactive computing stage.

Theorem 3.2 (Correctness of *FIRST-E*). *The Algorithm 3 (FIRST-E) gives the approximate updated similarity vector by Definition 1:* $\tilde{\mathbf{s}} = \hat{\mathbf{s}}$.

Proof We know that $\tilde{\mathbf{W}}$ takes the form of type C (which is $\mathbf{D}^{-1/2}\mathbf{N}[\mathbf{E} \odot (\mathbf{A} \otimes \mathbf{A_q})]\mathbf{N}\mathbf{D}^{-1/2}$). Then

$$\begin{aligned}\tilde{\mathbf{W}}^{\mathbf{sym}} &= \mathbf{D}^{-\frac{1}{2}}\mathbf{N}[(\sum_{l=1}^{L} \mathbf{E}^l_A \otimes \mathbf{E}^l_q) \odot (\mathbf{A} \otimes \tilde{\mathbf{A_q}})]\mathbf{N}\mathbf{D}^{-\frac{1}{2}} \\ &= \mathbf{D}^{-\frac{1}{2}}\mathbf{N}[\sum_{l=1}^{L} (\mathbf{E}^l_A \odot \mathbf{A}) \otimes (\mathbf{E}^l_q \odot \tilde{\mathbf{A_q}})]\mathbf{N}\mathbf{D}^{-\frac{1}{2}} \qquad (7)\end{aligned}$$

$$\begin{aligned}\hat{\mathbf{W}}^{\mathbf{sym}} &= \mathbf{D}^{-\frac{1}{2}}\mathbf{N}[\sum_{l=1}^{L} (\mathbf{U^l_A}\mathbf{\Lambda^l_A}(\mathbf{U^l_A})^\mathbf{T}) \otimes (\mathbf{U^l_q}\mathbf{\Lambda^l_q}(\mathbf{U^l_q})^\mathbf{T})]\mathbf{N}\mathbf{D}^{-\frac{1}{2}} \qquad (8) \\ &= \mathbf{D}^{-\frac{1}{2}}\mathbf{N}[\sum_{l=1}^{L} (\mathbf{U^l_A} \otimes \mathbf{U^l_q})(\mathbf{\Lambda^l_A} \otimes \mathbf{\Lambda^l_q})((\mathbf{U^l_A})^T \otimes (\mathbf{U^l_q})^T)]\mathbf{N}\mathbf{D}^{-\frac{1}{2}} \\ &= \mathbf{D}^{-\frac{1}{2}}\mathbf{N}\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathbf{T}\mathbf{N}\mathbf{D}^{-\frac{1}{2}}\end{aligned}$$

where $\mathbf{U}$ and $\mathbf{\Lambda}$ are block matrices as described in line 20 of Algorithm 3. From Equation 7 to Equation 8, the top r and top t eigenvalue decomposition are taken as described in line 6 and line 18. The derivation from the second line to the third line in Equation 8 is based on the property of block matrix [19]. If $\mathbf{l}'$ is not empty, in the interactive stage,

$$\begin{aligned}\hat{\mathbf{W}}^{\mathbf{sym}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{N}[&\sum_{l \in \mathbf{l}'} (\mathbf{U^l_A}\mathbf{\Lambda^l_A}(\mathbf{U^l_A})^\mathbf{T}) \otimes (\mathbf{U^l_q}\mathbf{\Lambda^l_q}(\mathbf{U^l_q})^\mathbf{T}) \\ &+ \sum_{l \notin \mathbf{l}'} (\mathbf{U^l_A}\mathbf{\Lambda^l_A}(\mathbf{U^l_A})^\mathbf{T}) \otimes (\mathbf{U^l_q}\mathbf{\Lambda^l_q}(\mathbf{U^l_q})^\mathbf{T})]\mathbf{N}\mathbf{D}^{-\frac{1}{2}} \qquad (9)\end{aligned}$$

where the eigen-decomposed term that is not in the index $\mathbf{l}'$ (in the second line of equation 9) is calculated in the precomputing stage (line 6). Still, equation 9 is equal to equation 8. Let $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{N}\mathbf{U}$, $\mathbf{R} = \mathbf{U}^\mathbf{T}\mathbf{N}\mathbf{D}^{-1/2}$. Then $\hat{\mathbf{s}}$ is given as:

$$\begin{aligned}\hat{\mathbf{s}} &= (1 - \alpha)(\mathbf{I} - \alpha\hat{\mathbf{W}})^{-1}\mathbf{h} \\ &= (1 - \alpha)(\mathbf{I} - \alpha\mathbf{L}\mathbf{\Lambda}\mathbf{R})^{-1}\mathbf{h} \\ &= (1 - \alpha)[\mathbf{I} + \alpha\mathbf{L}(\mathbf{\Lambda}^{-1} - \alpha\mathbf{R}\mathbf{L})^{-1}\mathbf{R}]\mathbf{h}\end{aligned}$$

The last line comes from Sherman-Morrison Lemma [20]. Hence we have proved that $\tilde{\mathbf{s}} = \hat{\mathbf{s}}$ and FIRST-E gives the approximate updated similarity vector. □

3.4 Implementation Details

In this section, we present implementation details of our method to transform the similarity matrix to one or more matching subgraphs. After either FIRST-Q, FIRST-N or FIRST-E is called, the returned similarity matrix $\tilde{\mathbf{S}}$ should be transfered into updated matching subgraph $\tilde{\mathcal{M}}$. We start by introducing indicator matrix $\mathbf{X}$ and "goodness" function.

Let $\mathbf{X}$ be a $k \times n$ binary match indicator matrix, where $\mathbf{X}(i, j) = 1$ means that the i^{th} query node in $\mathcal{Q}$ matches the j^{th} node in $\mathcal{G}$; and $\mathbf{X}(i, j) = 0$ otherwise. It can be seen that each row of $\mathbf{X}$ has only one entry to be 1 and each column of $\mathbf{X}$ has at most one entry to be 1. The match indicator matrix $\mathbf{X}$ induces the matching subgraphs, i.e., the matching subgraph are the induced subgraph of $\mathcal{G}$ whose corresponding columns in $\mathbf{X}$ are not empty. The match indicator matrix $\mathbf{X}$ can be found through the following "goodness" function [2].

$$g(X) = -\| \mathbf{XAX'} - \mathbf{A_q} \|_F^2 + a \cdot \text{trace}(\mathbf{SX'}) - b \cdot \| \mathbf{XX'} - \mathbf{I} \|_F^2$$

where a and b are parameters that balance the weight of each term.

The idea behind the goodness function to calculate $X(k \times n)$ that best embodies both the rankings in the pair-wise similarity matrix (the second term) and the original connectivity consistency (the first term of the network. In our proposed method, we first drive the index matrix $\mathbf{T}$, which gives the node pairs that should be connected in the resulting matching subgraph; then convert all connected subgraphs to "super nodes" (i.e., connected node sets); and finally find the bridges between the corresponding super nodes. The proposed procedure to find matching subgraph is summarized in Algorithm 4.

Algorithm 4 SIM2SUB

```
Input: Indicator Matrix $X(k \times n)$, the data network $\mathcal{G}$, revised query graph $\tilde{Q}$.
Output: The updated matching subgraph $\tilde{\mathcal{M}}$
 1: $\mathbf{T} = \mathbf{X'A_qX} - \text{and}(\mathbf{A}, \mathbf{X'A_qX})$; //index matrix of nodes in subgraph that
    should be connected;
 2: Construct index $I$ of connected nodes from $and(\mathbf{A}, \mathbf{X'A_qX})$;
 3: for each connected node set $C$ in $I$ do
 4:     Construct "super node" $C'$;
 5: end for
 6: Update the data network $\mathcal{G}$;
 7: for each unconnected "super node" pair ($C_1'$, $C_2'$) from T do
 8:     Connect ($C_1'$, $C_2'$) by the shortest path;
 9: end for
10: return $\tilde{\mathcal{M}}$;
```

Since the counterpart of the query graph can be found in the indicator matrix $\mathbf{X}$, the connection among the counterparts should be decided. In line 1 and 2, the algorithm finds the indices of nodes that are directly connected and not connected in the original data network $\mathcal{G}$. From line 3 to line 10, the algorithm constructs super nodes and find shortest paths as bridges among counterparts that should be connected. The method generates one subgraph from one indicator matrix $\mathbf{X}$. If multiple results (e.g., t results) are required, then t indicator matrices which have top t largest "goodness" values will be constructed as described in Section 3.1.

3.5 Complexity Analysis

In this section we give the complexity analysis of the proposed algorithms (i.e., FIRST-Q, FIRST-N and FIRST-E).

LEMMA 3.3. ***Complexity of FIRST-Q** & **FIRST-N**. The time complexity of Algorithm 1 and Algorithm 2 is $O(r^2t^2kn + rtkn + K^2kn)$, and its space complexity is $O(k^2rn + m_1)$. Here, n and k are the orders of the number of nodes of the input data network and the query graph, respectively; K denotes the number of unique node attributes, and r and t are the rank of eigendecomposition; m_1 is the number of edges in attributed network $\mathcal{G}$.*

PROOF Firstly, since the diagonal degree matrix $\mathbf{D}$ needs to be updated, from equation 2, $\mathbf{D}$ is given as follows when edge attribute is missing:

$$\mathbf{D} = \text{diag}(\sum_{k,k'=1}^{K} \underbrace{(\mathbf{N}^k\mathbf{A}\mathbf{N}^{k'}\mathbf{1}) \otimes \overbrace{(\mathbf{N}_q^k\mathbf{A}_q\mathbf{N}_q^{k'}\mathbf{1})}^{O(k^2)}}_{O(k^2)+O(nk)=O(nk))}$$

This is because $k \ll n$ based on our assumption. Also note that during the updating, $\mathbf{A}$ and $\mathbf{N}$ of the network $\mathcal{G}$ is unchanged. In all, the complexity of updating $\mathbf{D}$ is $O(K^2kn)$. In the process of computing $\tilde{\mathbf{s}}$,

$$\tilde{\mathbf{s}} = \underbrace{(1-\alpha)\mathbf{P}^{-1}[\mathbf{D_1}^{-1} + \alpha\mathbf{D_1}^{-1}\mathbf{L} \quad \overbrace{(\mathbf{\Lambda}^{-1} - \alpha\mathbf{R}\mathbf{D_1}^{-1}\mathbf{L})^{-1}}^{O(r^2t^2kn)+O(r^3)=O(r^2t^2kn)} \quad \mathbf{R}\mathbf{D_1}^{-1}]\mathbf{P}^{-1}\mathbf{h}}_{O(r^2t^2kn)+O(rtkn)}$$

Secondly, when computing the above multiplication in backward way in order to make use of the linear complexity property of vector multiplication, the complexity can achieve $O(r^2t^2kn)+O(rtkn)$. The time for the rest of the computation in the algorithm is smaller, and thus can be ignored in the big-O notation. Overall, the time complexity of FIRST-Q is $O(r^2t^2kn + rtkn + K^2kn)$.

The proof of space complexity is omitted for space. □

LEMMA 3.4. ***Complexity of FIRST-E**. The time complexity of Algorithm 3 is $O(r^2t^2kn + Lrtkn + K^2Lkn)$, and its space complexity is $O(Lrtkn + m_1)$. Here, n and k are the orders of the number of nodes of the input network and query graph, respectively; K, L denotes the number of unique node and edge attributes, respectively and r, t are the rank of eigendecomposition; m_1 is the number of edges in attributed network $\mathcal{G}$.*

PROOF When $\mathbf{N}$ and $\mathbf{E}$ are both available, in the process of updating diagonal degree matrix $\mathbf{D}$:

$$\mathbf{D} = \text{diag}(\underbrace{\sum_{k,k'=1}^{K}\sum_{l=1}^{L}(\mathbf{N}^k(\mathbf{E}^l \odot \mathbf{A})\mathbf{N}^{k'}\mathbf{1}) \otimes \overbrace{(\mathbf{N}_q^k(\mathbf{E}_q^l \odot \mathbf{A}_q)\mathbf{N}_q^{k'}\mathbf{1}))}^{O(k^2)}}_{O(K^2Lkn)}$$

$$\tilde{\mathbf{s}} = \underbrace{(1-\alpha)[\mathbf{I} + \alpha\mathbf{L} \quad \overbrace{(\mathbf{\Lambda}^{-1} - \alpha\mathbf{RL})^{-1}}^{O(Lrtkn)+O(L^3r^3t^3)+O(r^2t^2kn)} \quad \mathbf{R}]\mathbf{h}}_{O(r^2t^2kn+Lrtkn)}$$

As we see from the above equations with the heaviest computation in the algorithm, the time complexity of FIRST-E is $O(r^2t^2kn + Lrtkn + K^2Lkn)$. The complexity of the rest of the computation in the algorithm can be reasonably ignored.

The proof of space complexity is omitted for space. □

4 EXPERIMENTAL RESULTS

In this section, we present the experimental results and analysis of our proposed algorithms. The experiments are designed to answer the following questions:

(1) **Effectiveness** How effective are our proposed subgraph generating algorithms compared to other algorithms when different structures of queries are sent as inputs?
(2) **Efficiency** How fast are our proposed algorithms compared to other techniques when they are applied on different size of real networks? How do our algorithms scale?

[2]In this paper, we use a local heuristic to find $\mathbf{X}$ from $\mathbf{S}$ by searching the top-l entries in each row of $\mathbf{S}$, where l is a small number (e.g., $l = 3$).

4.1 Experimental Setup

Datasets. We use five different real-world datasets in our experiments, summarized in Table 2.

Table 2: Datasets Used in Evaluations

Name	# of Nodes	# of Edges	Node/Edge Attribute
DBLP	9,143	16,338	Node attribute only
Flickr	12,974	16,149	Node attribute only
LastFm	136,421	1,685,524	Node attribute only
ArnetMiner	1,274,360	4,756,194	Node & edge attribute
LinkedIn	6,726,290	19,360,690	Node attribute only

- *DBLP*: In the graph of *DBLP* each node represents an author who published paper in popular Data Mining and Database conferences and journals. Undirected edges represent co-authorship and each author has one attribute vector of the number of publications in 29 major conferences[22].
- *Flickr*: The graph of this dataset is the network of friends on *Flickr*. Node attribute vector is transformed from users' profile information [30].
- *LastFm:* This dataset contains the following relationships of users on *LastFm* [30]. The node attribute vector is also transformed from users' profile information, such as age, gender and location. It was collected in 2013.
- *ArnetMiner:* The graph in ArnetMiner dataset represents the academic social network. Undirected edges represent co-authorship and node attribute vector is extracted from number of published papers[30].
- *LinkedIn:* The graph of LinkedIn dataset is from users' connection relationship in the social network in *LinkedIn*. The node attribute vector is transformed from users' profile information such as age, gender and occupation, etc.

Comparison Methods. We compare our proposed algorithms with *G-Ray* [27], *MAGE* [21], *FINAL* together with its variants (e.g., *FINAL-N*, *FINAL-N+*, *FINAL-NE*) [29]. To be specific, according to the two-phase nature of our method, we compare the similarity matrix calculated by our method and *FINAL* and the subgraph with *G-Ray* and *MAGE*. We also verify the efficiency of our proposed method on five real-world datasets and test the scalability.

Machine. The following experiments are tested on 64-bit Windows Machine with 3.60 GHz CPU and 32.0 GB RAM. Programs are implemented in MATLAB with single thread.

4.2 Effectiveness

A - Matching Graph Comparison. We evaluate the effectiveness of each of the two phases of the whole algorithm. First, we show how well our algorithms can perform on computing the cross-network node similarities. We define the distance of two similarity matrices computed by two different methods. The distance is calculated as the Frobenius norm of the difference between two similarity matrices: $distance = \parallel \mathbf{S} - \mathbf{S}' \parallel_F$, where S is computed by *FINAL-N+* and S′ is computed by FIRST. The distance against the number of query nodes indicate the closeness between the similarity matrix returned from FIRST and *FINAL-N+*. As we observe from Figure 2, as the number of query nodes increase, the distance can be lower than 1.5×10^{-6}, which indicates that the similarity results computed by these two methods are quite close. Next we treat the similarity matrix returned from *FINAL-N+* as groundtruth and define the precision and recall to evaluate how well our method approximates *FINAL-N+*. First we sort both similarity matrix S and S′ and truncate top k columns as retrieved results **M**. The precision and recall are then calculated with regard to p ($p \leq k$) selected columns. For each row, if the entry in **S'** of selected matrix is also in **M**, then we consider it as relevant. From Figure 2, we can observe that the precision tends to be high when the recall is low. Also the overall tendency shows that a relatively small r leads to a high precision.

In the second phase, we test the effectiveness of our subgraph generating algorithm against *G-Ray* and *MAGE* respectively. We design five typical query patterns and load them into three algorithms. The performance of three algorithms are summarized in Table 4 and Table 5. We define the terminology in the table as follows.

According to the summary, the patterns returned by *G-Ray* deviate significantly from the query pattern. The returned subgraph is reasonable in several patterns such as E-star (%83.3 exact matching nodes) and line (%50 exact matching nodes). For other patterns like clique (FIRST %57.1 vs. *G-Ray* %25.0 exact matching nodes) and loop (FIRST %71.4 vs. *G-Ray* %27.3 exact matching nodes), our method performs better. Generally speaking, thanks to our formulation that considers both the topology and pairwise node similarity, when the inputs contain more diverse and complicate patterns, the results returned by our method tends to have a better balance on the subgraph structure and attribute matching. Overall, our method also outperforms *MAGE* when edge attribute is taken into consideration.

Table 3: Terminology Definition in Table 4 & 5

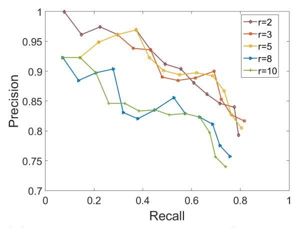
Name	Definition
Extra Nodes	Nodes in subgraph with incorrect node attribute or related position
Exact Matching Nodes	Nodes in subgraph with correct node attribute and related position
Intermediate Nodes	Nodes in the path between exact matching nodes
Extra Edges	Edges in subgraph with incorrect edge attribute or between extra nodes
Exact Matching Edges	Edges in subgraph with correct node attribute and between exact matching nodes
Intermediate Edges	Edges in the path between intermediate nodes

Table 4: Matching Comparison of 5 Patterns (Nodes)

	% Extra Nodes			% Exact Matching Nodes			% Intermediate Nodes		
Algorithm	G-Ray	MAGE	FIRST	G-Ray	MAGE	FIRST	G-Ray	MAGE	FIRST
Star(N)	62.5	*	**0.0**	37.5	*	**75.0**	0.0	*	25.0
E-Star(N)	**0.0**	*	**0.0**	**83.3**	*	71.4	16.7	*	28.6
Line(N)	50.0	*	**0.0**	50.0	*	**83.3**	0.0	*	16.7
Loop(N)	**0.0**	*	**0.0**	27.3	*	**71.4**	72.7	*	28.6
Clique(N)	25.0	*	**0.0**	25.0	*	**57.1**	50.0	*	42.9
Star(NE)	*	50.0	**0.0**	*	30.0	**40.0**	*	20.0	60.0
E-Star(NE)	*	**0.0**	**0.0**	*	33.3	**41.7**	*	66.7	58.3
Line(NE)	*	33.3	**0.0**	*	33.3	**62.5**	*	33.3	37.5
Loop(NE)	*	**27.3**	44.4	*	27.3	**33.3**	*	45.5	22.2
Clique(NE)	*	40.0	**0.0**	*	60.0	**66.7**	*	0.0	33.3

Table 5: Matching Comparison of 5 Patterns (Edges)

	% Extra Edges			% Exact Matching Edges			% Intermediate Edges		
Algorithm	G-Ray	MAGE	FIRST	G-Ray	MAGE	FIRST	G-Ray	MAGE	FIRST
Star(N)	66.7	*	**0.0**	33.3	*	**57.1**	0.0	*	42.9
E-Star(N)	**0.0**	*	**0.0**	**60.0**	*	50.0	40.0	*	50.0
Line(N)	60.0	*	**0.0**	40.0	*	**60.0**	0.0	*	40.0
Loop(N)	8.3	*	**0.0**	8.3	*	**42.9**	83.3	*	57.1
Clique(N)	35.7	*	**0.0**	7.1	*	**12.5**	64.3	*	87.5
Star(NE)	*	42.9	**0.0**	*	0.0	**14.3**	*	57.1	85.7
E-Star(NE)	*	**0.0**	**0.0**	*	7.1	**9.0**	*	92.9	91.0
Line(NE)	*	27.3	**0.0**	*	0.0	**14.3**	*	72.7	85.7
Loop(NE)	*	**30.0**	42.9	*	0.0	**14.3**	*	70.0	29.8
Clique(NE)	*	33.3	**0.0**	*	0.0	**12.5**	*	100.0	87.5


(a) Precision vs. Recall ($\alpha = 0.8$).

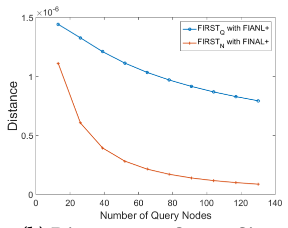
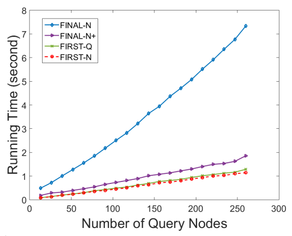

(b) Distance vs. Query Size.

Figure 2: Effectiveness Comparison (number of query nodes = 13, $r = 5$).


(a) Running time vs. Query Size comparison between *FINAL* and *FIRST*. ($\alpha = 0.8$).

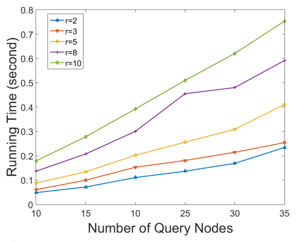
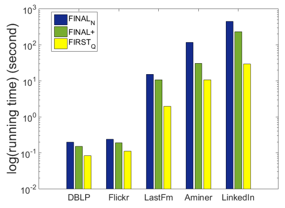

(b) Running time vs. Query Size and Eigendecompositon Rank. ($\alpha = 0.8$).

Figure 3: Scalability Comparison and Analysis on Query Size and Eigendecomposition Rank.

B - Case Study. We also designed an interactive scenario which is shown in Figure 5. The experiment is conducted on *DBLP* dataset. The initial query is a 4-node clique which requests a group of co-authors from four different main conferences. The initial result gives an approximate clique with three intermediate authors. After that, the user adds one more author from SIGKDD to the group and the algorithm returns an approximate subgraph with six intermediate authors. Note that as the query is refined, the result is also incrementally refined instead of a complete change, as Alan F. McMichael appears in both initial and updated result. This phenomena can be also observed from the following steps and it makes sense in the interactive procedure. As the user realizes that the structure is complex and incurs too many intermediate nodes, the query graph is revised to a loop fashion. The refined result shows H. Schweitzer and Enrico Motta in both results. Finally the user is shown a refined output with five intermediate nodes, and all five matching nodes are exact matching nodes from *DBLP*. This implies the cooperation pattern among the authors.

4.3 Efficiency

A - Speedup. We first evaluate the efficiency of our proposed technique and there are two phases in this particular experiment. First, we only consider node attribute and perform FIRST against *FINAL-N* and *FINAL-N+* to compare the running time on five datasets. Then we add edge attribute and perform FIRST against *FINAL-NE*. In each test, the query graph is fixed and it is a relatively small graph with 13 nodes. The results are shown in Figure 4. From the result we can observe that our algorithm outperforms the other three algorithms with speedup from 2× to 16×. Specifically, when the large graph has over 20M edges, The response time for exact network alignment method is too long to measure. But our method can incrementally update pair-wise similarity in about 20 seconds with a high accuracy (over 90%, see Section 4.2).


(a) Efficiency Comparison 1 ($\alpha = 0.8$).

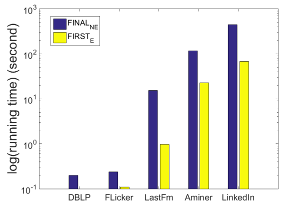

(b) Efficiency Comparison 2 ($\alpha = 0.8$).

Figure 4: (Lower is better.) Log of Running time vs Datasets of different size. (number of query nodes = 13, $r = 5$).

B - Scalability. The scalability of FIRST is summarized in Figure 3 and they are tested on *DBLP*. We first measure the running time against the number of query nodes and also compare it with *FINAL-N* and *FINAL-N+*. We can see that the running time of FIRST grows linearly. Compared to *FINAL-N* and *FINAL-N+*, FIRST has better scalability as the increase of query size. Next we measure the scalability with regard to the number of eigenvalues used in FIRST. We can see that for different r, the running time still grows linearly. At the point where there are 60 nodes and $r = 10$, the running time is still less than 1 second (0.8s). It shows that the quick response time of FIRST, which makes it suitable for interactive query feedback.

5 RELATED WORK

Various subgraph matching techniques can be found in different targeting problems. G-Ray by Tong et al. [27] applies *RWR* (Random Walk with Restart) [18, 28] and *CePS* (CenterPiece Subgraphs) [26] idea to achieve fast inexact pattern matching for networks with node attributes. *TALE* by Tian and Patel [25] allows approximate matching and large query graphs by proposing NH-index (Neighborhood Index). *SIGMA* by Mongiovi et al. [17] defines a new effective pruning rule for inexact matching based on multi-set and multi-cover, a variant of the well known set-cover problem. It performs well in the application of query yeast and human protein complexes. More recently, *R-WAG*, *I-WAG* and *S-WAG* by Roy et al. [23] aim to return fast best-effort answer for WAG(Weighted Attributed Graphs) query by designing a hybrid index structure that incorporates weighted attributes, structural features and graph structure. *NeMa* by Khan et al. [12] proposes a heuristic approach based on defining a new definition of matching cost metric. More recently, *IncMatch* by Fan et al. presents a simulated method for incremental subgraph matching of certain patterns.

One of the important aspects of our method is to measure the similarity between nodes. Some works adopt similar methodology (compared to our technique) to address dynamic issues. While the *SimRank* by G. Jeh [11] is well known for its effectiveness, recently the fast algorithms by C. Li et al. [14] is proposed. By using Kronecker product and vectorization operators, an approximate non-iterative approach is developed for similarity tracking in evolving information networks. Other works like *ObjectRank* [5], *RelationalRank* [9], direction-aware proximity [27] are all on measuring the node similarity. *COSNET* by Zhang et al. [30] connects

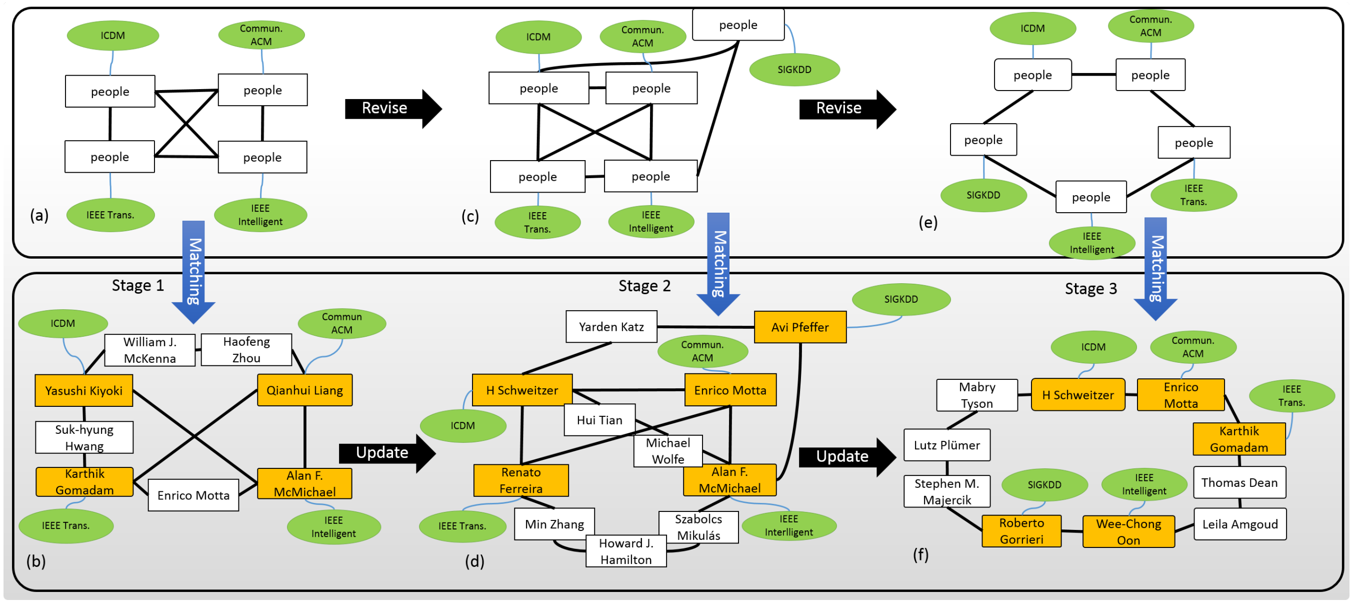


Figure 5: A case study of interactive attributed subgraph matching on *DBLP*. Best viewed in color. (a): query graphs, (b): matching subgraphs. Green ellipse: node attribute value, Yellow rectangle: matching node, White rectangle: extra/intermediate node.

heterogeneous social networks with local and global consistency, which lays the foundation of the idea of our method in filtering the Kronecker graph with node and edge attributes. Another common technique which is adopted in our method is low-rank approximation [1, 7]. Many real-world network has low-rank property and low-rank approximation is widely used in various data mining tasks. As for the usage of block matrix approaches in large graph mining, a large amount of applications could be found in the literature ranging from group/community detection [8] to fast algorithms for calculating *RWR* [24, 28].

Our fast and interactive approach has various potential applications such as dense subgraph detection [3, 13] in static [10]/dynamic graphs [4], large graph sense-making [2] and rare category detection [31]/ outlier detection/anomaly detection [32]. These works are closely related to pattern matching/recognition problems, which could be addressed as subgraph matching problems.

6 CONCLUSION

In this paper, we study the interactive attributed subgraph matching problem and develop a family of efficient and effective algorithms (FIRST) to address this problem according to different interactive scenarios. Specifically, we first propose that the problem can be recasted to a cross-netwrok node similarity problem and the computation can be speeded up by exploring the smoothness between initial and revised queries. We then propose FIRST-Q and FIRST-N to handle the scenario where only node attribute is available, and FIRST-E to handle the scenario where both node and edge attribute are available. We conduct numerous experiments on real world data, and show that our method lead up to 16× speedup with more than 90% accuracy. In the future, we will (i) deploy the proposed FIRST algorithms in an online team search and optimization system (http://team-net-work.org/system.html), and (ii) generalize it to handle dynamic attributed data networks and deploy it.

7 ACKNOWLEDGEMENTS

This work is supported by National Science Foundation under Grant No. IIS-1651203, DTRA under the grant number HDTRA1-16-0017, Army Research Office under the contract number W911NF-16-1-0168, National Institutes of Health under the grant number R01LM011986, Region II University Transportation Center under the project number 49997-33 25, National Natural Science Foundation of China under Grant No. 61602306, IBM 2016 SUR Award, and a Baidu gift. We would also like to genuinely thank Dr. Yutao Zhang and Dr. Jie Tang for sharing the dataset, and all the reviewers for providing helpful criticism and valuable comments.

REFERENCES

[1] Dimitris Achlioptas and Frank McSherry. 2007. Fast computation of low-rank matrix approximations. *Journal of the ACM (JACM)* 54, 2 (2007), 9.
[2] Leman Akoglu, Duen Horng Chau, Jilles Vreeken, Nikolaj Tatti, Hanghang Tong, and Christos Faloutsos. 2013. Mining connection pathways for marked nodes in large graphs. In *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 37–45.
[3] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery* 29, 3 (2015), 626–688.
[4] Albert Angel, Nikos Sarkas, Nick Koudas, and Divesh Srivastava. 2012. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *Proceedings of the VLDB Endowment* 5, 6 (2012), 574–585.
[5] Andrey Balmin, Vagelis Hristidis, and Yannis Papakonstantinou. 2004. Objectrank: Authority-based keyword search in databases. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 564–575.
[6] Nan Cao, Yu-Ru Lin, Liangyue Li, and Hanghang Tong. 2015. g-Miner: Interactive visual group mining on multivariate graphs. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 279–288.
[7] Petros Drineas, Ravi Kannan, and Michael W Mahoney. 2006. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix.

SIAM Journal on computing 36, 1 (2006), 158–183.

[8] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3 (2010), 75–174.

[9] Floris Geerts, Heikki Mannila, and Evimaria Terzi. 2004. Relational link-based ranking. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 552–563.

[10] David Gibson, Ravi Kumar, and Andrew Tomkins. 2005. Discovering large dense subgraphs in massive graphs. In *Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 721–732.

[11] Glen Jeh and Jennifer Widom. 2002. SimRank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 538–543.

[12] Arijit Khan, Yinghui Wu, Charu C Aggarwal, and Xifeng Yan. 2013. Nema: Fast graph search with label similarity. In *Proceedings of the VLDB Endowment*, Vol. 6. VLDB Endowment, 181–192.

[13] Victor E Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. 2010. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*. Springer, 303–336.

[14] Cuiping Li, Jiawei Han, Guoming He, Xin Jin, Yizhou Sun, Yintao Yu, and Tianyi Wu. 2010. Fast computation of simrank for static and dynamic information networks. In *Proceedings of the 13th International Conference on Extending Database Technology*. ACM, 465–476.

[15] Liangyue Li, Hanghang Tong, Nan Cao, Kate Ehrlich, Yu-Ru Lin, and Norbou Buchler. 2015. Replacing the irreplaceable: Fast algorithms for team member recommendation. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 636–646.

[16] Liangyue Li, Hanghang Tong, Nan Cao, Kate Ehrlich, Yu-Ru Lin, and Norbou Buchler. 2016. TEAMOPT: Interactive Team Optimization in Big Networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2485–2487.

[17] Misael Mongiovi, Raffaele Di Natale, Rosalba Giugno, Alfredo Pulvirenti, Alfredo Ferro, and Roded Sharan. 2010. Sigma: a set-cover-based inexact graph matching algorithm. *Journal of bioinformatics and computational biology* 8, 02 (2010), 199–218.

[18] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. 2004. Automatic multimedia cross-modal correlation discovery. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 653–658.

[19] Kaare Brandt Petersen, Michael Syskind Pedersen, and others. 2008. The matrix cookbook. *Technical University of Denmark* 7 (2008), 15.

[20] Walter W Piegorsch and George Casella. 1990. Erratum: inverting a sum of matrices. *SIAM review* 32, 3 (1990), 470–470.

[21] Robert Pienta, Acar Tamersoy, Hanghang Tong, and Duen Horng Chau. 2014. Mage: Matching approximate patterns in richly-attributed graphs. In *Big Data (Big Data), 2014 IEEE International Conference on*. IEEE, 585–590.

[22] Adriana Prado, Marc Plantevit, Céline Robardet, and Jean-Francois Boulicaut. 2013. Mining graph topological patterns: Finding covariations among vertex descriptors. *Knowledge and Data Engineering, IEEE Transactions on* 25, 9 (2013), 2090–2104.

[23] Senjuti Basu Roy, Tina Eliassi-Rad, and Spiros Papadimitriou. 2015. Fast best-effort search on graphs with multiple attributes. *IEEE Transactions on Knowledge and Data Engineering* 27, 3 (2015), 755–768.

[24] Kijung Shin, Jinhong Jung, Sael Lee, and U Kang. 2015. Bear: Block elimination approach for random walk with restart on large graphs. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 1571–1585.

[25] Yuanyuan Tian and Jignesh M Patel. 2008. Tale: A tool for approximate large graph matching. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*. IEEE, 963–972.

[26] Hanghang Tong and Christos Faloutsos. 2006. Center-piece subgraphs: problem definition and fast solutions. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 404–413.

[27] Hanghang Tong, Christos Faloutsos, Brian Gallagher, and Tina Eliassi-Rad. 2007. Fast best-effort pattern matching in large attributed graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 737–746.

[28] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. (2006).

[29] Si Zhang and Hanghang Tong. 2016. Final: Fast attributed network alignment. In *Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM*.

[30] Yutao Zhang, Jie Tang, Zhilin Yang, Jian Pei, and Philip S Yu. 2015. Cosnet: Connecting heterogeneous social networks with local and global consistency. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1485–1494.

[31] Dawei Zhou, Jingrui He, K Seluk Candan, and Hasan Davulcu. 2015. MUVIR: Multi-View Rare Category Detection.. In *IJCAI*. 4098–4104.

[32] Dawei Zhou, Kangyang Wang, Nan Cao, and Jingrui He. 2015. Rare category detection on time-evolving graphs. In *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 1135–1140.