ATTENT: Active Attributed Network Alignment

Qinghai Zhou*, Liangyue Li†, Xintao Wu‡, Nan Cao§, Lei Ying⋆, Hanghang Tong*

*University of Illinois at Urbana-Champaign, {qinghai2, htong}@illinois.edu;

†Alibaba Group, liliangyue.lly@alibaba-inc.com;

‡University of Arkansas, xintaowu@uark.edu;

§Tongji University, nan.cao@gmail.com;

⋆University of Michigan, Ann Arbor, leiying@umich.edu

ABSTRACT

Network alignment finds node correspondences across multiple networks, where the alignment accuracy is of crucial importance because of its profound impact on downstream applications. The vast majority of existing works focus on how to best utilize the topology and attribute information of the input networks as well as the anchor links when available. Nonetheless, it has not been well studied on how to boost the alignment performance through actively obtaining high-quality and informative anchor links, with a few exceptions. The sparse literature on active network alignment introduces the human in the loop to label some seed node correspondence (i.e., anchor links), which are informative from the perspective of querying the most uncertain node given few potential matchings. However, the direct influence of the intrinsic network attribute information on the alignment results has largely remained unknown. In this paper, we tackle this challenge and propose an active network alignment method (ATTENT) to identify the best nodes to query. The key idea of the proposed method is to leverage effective and efficient influence functions defined over the alignment solution to evaluate the goodness of the candidate nodes for query. Our proposed query strategy bears three distinct advantages, including (1) *effectiveness*, being able to accurately quantify the influence of the candidate nodes on the alignment results; (2) *efficiency*, scaling linearly with $15-17\times$ speed-up over the straightforward implementation without any quality loss; (3) *generality*, consistently improving alignment performance of a variety of network alignment algorithms.

ACM Reference Format:

Qinghai Zhou*, Liangyue Li†, Xintao Wu‡, Nan Cao§, Lei Ying⋆, Hanghang Tong*. 2021. ATTENT: Active Attributed Network Alignment. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3442381.3449886

1 INTRODUCTION

Network alignment finds node correspondences across multiple networks, where the alignment accuracy is paramount as it has a profound impact on the performance of downstream applications, ranging from aligning identical users across social networks [1, 29], identifying the same customers in different transaction networks for financial fraud detection [27], comparing the similarity of two protein-protein interaction (PPI) networks by pairing homologous proteins or modules inside the networks [6], to matching similar objects in computer vision [37].

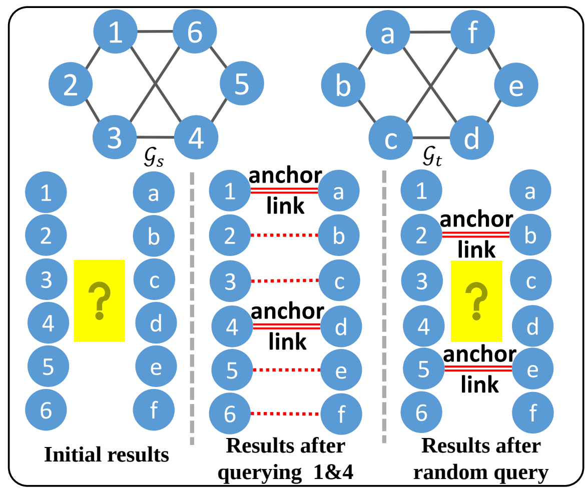


Figure 1: An illustrative example of active attributed network alignment. The red dashed line represents alignment between two nodes and the double red solid line is an anchor link. Given two network $\mathcal{G}_s$ and $\mathcal{G}_t$, the correct alignment is difficult to be determined at the beginning. For example, node 2 could be aligned to either b or e. By querying nodes $1, 4$ and obtaining their correct matching a, d, respectively (i.e., anchor links between 1 and a, 4 and d), all the remaining nodes can then be correctly aligned according to topological consistency. However, if we randomly query two nodes, e.g., nodes $2, 5$, then the correct matching for nodes $1, 3, 4, 6$ still cannot be determined. For example, it is possible that node 1 is aligned to either a or c because of graph isomorphism.

Known as a constrained quadratic assignment problem of NP-hardness, network alignment's applicability is often limited by its alignment accuracy. To improve the alignment accuracy, the vast majority of existing works focus on how to utilize the topology as well as attribute information of the input networks [2, 28, 30]. Nonetheless, it has been underexamined on how to *actively* obtain high-quality anchor links to further improve the alignment accuracy for a broad range of network alignment algorithms has not been well studied, with a few exceptions.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449886

The sparse literature on active network alignment addresses this problem when some anchor links become available by introducing a human annotator to label the correct matching for a given query [17]. And the selection of query nodes is determined by the marginal probability distribution (i.e., the certainty in finding the true alignment). For example, given some sampled alignment results, if a node is always aligned to the same node in the other network, it is deemed a highly certain alignment. Therefore this method [17] aims to query the nodes with high uncertainty. However, there are three major limitations with the existing query strategy. First, the direct influence of the query on the alignment result is not well investigated because the selection of nodes to query depends on sampled results of alignment. Second, how the network attribute information can improve the query strategy in active network alignment is not intensively studied. Third, it might bear a high computational complexity, depending on the specific sampling strategy (e.g., *TopMatchings* [17]).

In this paper, we address these limitations and propose an influence function-based query strategy (Attent) for active attributed network alignment according to the impact of query on the network alignment results, to suggest the best query to human annotator to label the correct alignment. Figure 1 presents an illustrative example of active attributed network alignment. The key idea of the proposed query method is to leverage the influence function defined over the network alignment results to evaluate the goodness of the candidate nodes to query. To be specific, given the alignment solution vector from the attributed network alignment algorithm [30], we quantify the query's informativeness as the rate of change of our proposed utility function over the alignment results w.r.t. the query. We further propose efficient algorithms to speed up and scale up the computation.

We summarize the main contributions of this paper as follows,

- **Problem Formulation.** We formally define the active attributed network alignment problem. The key idea is to measure the query's influence as the rate of change of our proposed utility function over network alignment results w.r.t. the query.
- **Algorithms and Analysis.** We propose a family of query algorithms (Attent) based on influence functions to solve the active attributed network alignment problem, which is applicable to a variety of network alignment algorithms. We further propose efficient algorithms to speed up and scale up the computations, with linear complexity.
- **Empirical Evaluations.** We conduct extensive experimental evaluations on real-world datasets to test the efficacy of our proposed method. The evaluation results demonstrate (1) *effectiveness*, the proposed methods outperform all baseline methods in achieving higher alignment accuracy; (2) *efficiency*, the proposed fast solutions scale linearly w.r.t. the network size, with $15-17\times$ speed-up over straight-forward implementations without any quality loss; and (3) *generality*, the proposed query strategies work effectively for a variety of network alignment algorithms.

The rest of the paper is organized as follows. We first review related work in Section 2. In Section 3, we define the problem of active attributed network alignment. We introduce the technical details in Section 4. Section 5 presents the experimental evaluation results. We finally conclude this paper in Section 6.

2 RELATED WORK

In this section, we review the related work in terms of (1) network alignment, and (2) active learning.

Network Alignment. Network alignment has attracted extensive research interests due to its broad applications in social network analysis [35], common protein molecules identification [6, 23], financial fraud detection [27], etc. One of the earliest work is *IsoRank* [24], computed in a PageRank-like way with the intuition that one node in a network is a good match for a node in another network if the former's neighbors are good matches for the latter's neighbors. *NetAlign* [2] formulates the network alignment problem as an integer quadratic problem that maximizes a linear combination of matching weights and the number of squares and finds the near-optimal solution by a distributed message-passing approach. These methods are solely based on topological information, which might be insufficient to deal with the inherent node or edge ambiguities. *FINAL* [30] instead augments the alignment consistency principle to include both node and edge attributes, in addition to network topology. In addition to finding the node correspondence across different networks, *MOANA* [32] finds the alignment among the network clusters at different granularities.

Following the success of recent network representation learning work, network alignment in the embedding space is gaining attractions. *REGAL* [11] embeds the nodes through a cross-network matrix factorization approach without explicitly constructing the full similarity matrix and employs a k-d tree data structure for efficient node alignment. *ORIGIN* [34] learns the cross-network embeddings through a Multi-GCN framework and aligns the nodes via a multi-view point set alignment approach. [15] proposes an embedding-based approach for mapping users across multiple networks by explicitly modeling each user's follower-ship and followee-ship. PALE [18] supervises the embedding with the known anchor links such that two latent spaces are non-linearly correlated. For multi-network alignment on different types of networks, CrossMNA [5] leverages the cross-network information to further refine the node representations. NetTrans [33] formulates the network alignment problem as a nonlinear process of transforming one network to another. Recent study also suggests that network alignment algorithm is vulnerable to adversarial perturbations [38].

Active Learning. In active learning, an oracle is asked to label some query instances so that the learning algorithm can achieve high accuracy with only a few labeled samples [8, 9, 13, 26]. Beyond i.i.d. data, the idea of active learning has also been applied to networks. For instance, criteria based on uncertainty, impact, and redundancy are used to select a batch of query nodes for labeling so as to maximize the network classification performance [22]. [3] proposes to exploit the interaction between the local and collective aspects of a classifier for selecting informative examples for better collective classification. The performance of semi-supervised graph embedding algorithms can also be optimized by actively selecting the nodes for labeling [4]. Integrating deep neural networks and adversarial learning with active learning, the divergence of unlabeled data and labeled ones is used for selecting the query nodes [14]. Early work on active network alignment is studied in [7]. *TopMatchings* and *GibbsMatchings* [17] are the two most relevant active network alignment methods based on bipartite matching, where the least uncertain nodes by sampling the matchings are queried. *ActiveIter* [20] addresses alignment in a different setting, namely heterogeneous networks, where it defines inter-network meta diagram for anchor link feature extraction and adopts active learning for effective label query.

3 PROBLEM DEFINITION

In this section, we first introduce the notations and preliminaries on attributed network alignment, and then we formally define the active attributed network alignment problem.

3.1 Notations

The main symbols and notations used in this paper are summarized in Table 1. We use bold uppercase letters for matrices (e.g., $\mathbf{A}$), bold lowercase letters for vectors (e.g., $\mathbf{h}$), uppercase letters for sets (e.g., V) and lowercase letters for scalars (e.g., k). To index a vector/matrix, we use $\mathbf{x}_i$ to denote the i^{th} element in the vector $\mathbf{x}$, $\mathbf{x}_{i:j}$ to represent the successive entries from $\mathbf{x}_i$ to $\mathbf{x}_j$, $\mathbf{A}(i,:)$ to denote the i^{th} row of $\mathbf{A}$, $\mathbf{A}(:,j)$ to denote the j^{th} column of $\mathbf{A}$, and $\mathbf{A}(i,j)$ to represent the entry at the i^{th} row and the j^{th} column of matrix $\mathbf{A}$.

Throughout the paper, we mainly focus on a pair of attributed source and target networks with the same node attribute dimension (i.e., K) and the same edge attribute dimension (i.e., L), represented as $\mathcal{G}_s = \{\mathbf{A}_s, \mathbf{N}_s, \mathbf{E}_s\}$ and $\mathcal{G}_t = \{\mathbf{A}_t, \mathbf{N}_t, \mathbf{E}_t\}$, respectively, where for each network $\mathcal{G}_i (i = s, t)$ with n_i nodes (1) $\mathbf{A}_i \in \mathbb{R}^{n_i \times n_i}$ is the adjacency matrix, (2) $\mathbf{N}_i \in \mathbb{R}^{n_i \times K}$ is the node attribute matrix, we denote the diagonalized j^{th} column of $\mathbf{N}_i$, i.e., $\text{diag}(\mathbf{N}_i(:,j))$ $(j = 1, \ldots, K)$, as $\mathbf{N}_i^j \in \mathbb{R}^{n_i \times n_i}$, which represents the values of the j^{th} attribute that all nodes have, and (3) $\mathbf{E}_i \in \mathbb{R}^{n_i \times n_i}$ is the edge attribute matrix and $\mathbf{E}_i^j \in \mathbb{R}^{n_i \times n_i}$ denotes the j^{th} edge attribute value that the edges have. The uppercase bold letters, $\mathbf{A}_\times, \mathbf{N}_\times, \mathbf{E}_\times \in \mathbb{R}^{n_s n_t \times n_s n_t}$ are the combined adjacency, node and edge attribute matrices, respectively, where $\mathbf{A}_\times = \mathbf{A}_s \otimes \mathbf{A}_t$, $\mathbf{N}_\times = \sum_{j=1}^{K} \mathbf{N}_s^j \otimes \mathbf{N}_t^j$, and $\mathbf{E}_\times = \sum_{j=1}^{L} \mathbf{E}_s^j \otimes \mathbf{E}_t^j$. For simplicity, we assume that the source and target networks are (a) unweighted and (b) undirected. It is straightforward to generalize the proposed method to weighted and/or directed networks of different sizes.

3.2 Preliminaries

In this subsection, we briefly review the optimization based attributed network alignment.

Attributed network alignment. The key idea behind aligning two input networks lies in the principle of consistency in (1) *topology*, (2) *node attribute*, and (3) *edge attribute* [31]. For example, considering two pairs of aligned nodes: (1) a_s in $\mathcal{G}_s$ and a_t in $\mathcal{G}_t$; and (2) b_s in $\mathcal{G}_s$ and b_t in $\mathcal{G}_t$, if a_s and b_s are close neighbors in $\mathcal{G}_s$, a_t and b_t should also be close neighbors in $\mathcal{G}_t$, i.e., *topology consistency*; a_s and a_t, b_s and b_t should share the same or similar node attribute values respectively, i.e., *node attribute consistency*; and edge (a_s, b_s) and (a_t, b_t) should also share the same or similar edge attribute value, i.e., *edge attribute consistency*. From the optimization's perspective, the attributed network alignment aims to find a solution matrix $\mathbf{X} \in \mathbb{R}^{n_t \times n_s}$ representing the cross-network node similarity to minimize the inconsistency in the previous three aspects and the vectorization of the solution matrix, i.e., $\mathbf{x} = \text{vec}(\mathbf{X})$ is obtained by solving the following equation,

Table 1: Symbols and Definition

Symbols	Definition
$\mathcal{G} = \{\mathbf{A}, \mathbf{N}, \mathbf{E}\}$	an attributed network
V, E	sets of vertices and edges in $\mathcal{G}$, respectively
$\mathbf{A}$	adjacency matrix
$\mathbf{N}^l$	diagonal matrix of the l^{th} node attribute
$\mathbf{E}^m$	matrix of the m^{th} edge attribute
$\mathbf{A}_\times, \mathbf{N}_\times, \mathbf{E}_\times$	combined adjacency, node and edge attribute matrix, respectively
$\mathbf{A}^{-1}, \mathbf{A}'$	inverse and transpose of matrix $\mathbf{A}$
$\mathbf{H}$	$n_t \times n_s$ prior alignment knowledge
$\mathbf{I}$	an identity matrix
q_{ij}, q_i	query a node pair (i, j) and a node i, respectively
C_i	set of candidate matchings of node i
$\mathcal{M}(i)$	matching of node i given alignment result $\mathcal{M}$
K, L	dimension of node and edge attributes, respectively
k, b	query budget, batch size
n, m	number of nodes and edges, respectively
$\otimes, \odot$	Kronecker product, entry-wise matrix product
$\mathbf{a} = \text{vec}(\mathbf{A})$	vectorize a matrix $\mathbf{A}$ in column order
$\mathbf{X} = \text{mat}(\mathbf{x}, n_t, n_s)$	reshape $\mathbf{x}$ to an $n_t \times n_s$ matrix in column order
$\mathbf{Y} = \text{diag}(\mathbf{y})$	diagonalize a vector $\mathbf{y}$

$$\mathbf{x} = \alpha \widetilde{\mathbf{W}} \mathbf{x} + (1 - \alpha)\mathbf{h} \quad (1)$$

where $\mathbf{h} = \text{mat}(\mathbf{H}, n_t, n_s)$, $\mathbf{H} \in \mathbb{R}^{n_t \times n_s}$ is the alignment preference matrix representing the preferred node similarity across two input networks, $\alpha \in (0, 1)$ is a regularization parameter and determines how close the alignment result vector $\mathbf{x}$ is to the preference vector $\mathbf{h}$, $\widetilde{\mathbf{W}}$ is the matrix which encodes the consistency in *topology, node attribute* and *edge attribute*, and $\widetilde{\mathbf{W}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}$ is the normalization of $\mathbf{W} = \mathbf{N}_\times (\mathbf{E}_\times \odot \mathbf{A}_\times) \mathbf{N}_\times$. The diagonal matrix $\mathbf{D}$ for normalizing $\mathbf{W}$ is defined as, $\mathbf{D} = \mathbf{N}_\times \text{diag}(\sum_{i=1}^{K} \sum_{j=1}^{L} [(\mathbf{E}_s^j \odot \mathbf{A}_s)\mathbf{N}_s^i] \otimes [(\mathbf{E}_t^j \odot \mathbf{A}_t)\mathbf{N}_t^i])$. Thanks to the property of the Kronecker product (i.e., $\text{vec}(\mathbf{ABC}) = (\mathbf{C}' \otimes \mathbf{A})\text{vec}(\mathbf{B})$), the solution vector $\mathbf{x}$ can be computed by an efficient iterative method using the following equation [30],

$$\mathbf{x} = \alpha \mathbf{D}^{-\frac{1}{2}} \mathbf{N}_\times \text{vec}\Big(\sum_{j=1}^{L} (\mathbf{E}_t^j \odot \mathbf{A}_t) \mathbf{Q} (\mathbf{E}_s^j \odot \mathbf{A}_s)'\Big) + (1 - \alpha)\mathbf{h} \quad (2)$$

which is equivalent to Eq. (1) and $\mathbf{Q}$ is a matrix after reshaping $\mathbf{q} = \mathbf{N}_\times \mathbf{D}^{-\frac{1}{2}} \mathbf{x}$ in the column order. If the node and edge attribute information is absent, Eq. (1) degenerates to $\mathbf{x} = \alpha \mathbf{D}_n^{-\frac{1}{2}} \mathbf{A}_\times \mathbf{D}_n^{-\frac{1}{2}} \mathbf{x} + (1-\alpha)\mathbf{h}$, where $\mathbf{D}_n = \text{diag}([\mathbf{A}_s \mathbf{1}] \otimes [\mathbf{A}_t \mathbf{1}])$ and $\mathbf{1}$ is an all-ones vector.

The alignment preference matrix, i.e., $\mathbf{H}$, contains prior knowledge of how similar two nodes across the two input networks are. For example, for node p, q in network $\mathcal{G}_s, \mathcal{G}_t$, respectively, if we set the value $\mathbf{H}(q, p) = 1$ and $\mathbf{H}(q, r)|_{r \neq p} \ll 1$ (i.e., nodes q and p are the correct alignment according to our prior knowledge), we then can obtain the solution matrix $\mathbf{X}$ where $\mathbf{X}(q, p) > \mathbf{X}(q, r)|_{r \neq p} = 0$. By the greedy matching method, we further conclude node q is aligned to node p, which is consistent with our preference. In the active learning setting of attributed network alignment, a human annotator is introduced to provide some prior information of correct alignment, which will be used to update the preference matrix $\mathbf{H}$ in order to maximally improve the performance of network alignment algorithms.

3.3 Problem Definition

Generally speaking, the goal of active learning is to maximally increase the learning performance by labeling as few samples from the whole training data as possible. In the case of active attributed network alignment, we assume that the algorithm can interact with an oracle (e.g., a human annotator) that can provide the feedback of desired information, i.e., the correct alignment between two networks. Throughout this paper, we mainly consider that the human annotator could answer the following question: *given a node a_s in the source network $\mathcal{G}_s$, and a set of candidate matchings, C_{a_s}, in the target network $\mathcal{G}_t$, which node a_t in C_{a_s} is the correct alignment of a_s?*

Given the vector $\mathbf{x}$ representing the solution of network alignment, we formally define the active attribute network alignment problem as follows,

Problem 1. Active Attributed Network Alignment

Given: (1) a pair of attributed source and target networks, $\mathcal{G}_s = \{\mathbf{A}_s, \mathbf{N}_s, \mathbf{E}_s\}$ and target network $\mathcal{G}_t = \{\mathbf{A}_t, \mathbf{N}_t, \mathbf{E}_t\}$, respectively, (2) an initial alignment preference matrix $\mathbf{H}_0$, (3) an integer query budget k, (4) an oracle, (5) the alignment result vector $\mathbf{x}$ from Eq. (2), (6) a utility function $f(\cdot)$;

Find: a set of k nodes in $\mathcal{G}_s$ which are most influential to $f(\mathbf{x})$, for the human annotator to label the correct matching in the target network $\mathcal{G}_t$ so that the alignment accuracy for the remaining nodes in $\mathcal{G}_s$ can be maximally improved.

4 ALGORITHMS AND ANALYSIS

In this section, we propose a quantitative method for evaluating the informativeness of the nodes based on the influence of nodes to the resulting alignment solution. Then we introduce our proposed query strategy as described in a generic algorithm for solving the active attributed network alignment problem, together with some analysis in effectiveness and efficiency.

4.1 Informativeness of Candidate Query

In active learning, a commonly used strategy is to select the data points that are more informative in training a machine learning model, e.g., leveraging uncertainty to evaluate the informativeness of query and selecting the data points with the most considerable uncertainty [21]. The main objective of active attributed network alignment is to find an effective query strategy to label the matched nodes in another network, so that the network alignment accuracy for the remaining nodes can be maximally improved. To achieve an effective query for network alignment, it is desirable to select nodes whose query feedback would significantly impact the alignment results, i.e., $\mathbf{x}$ in Eq. (1). The choices of the utility function $f(\cdot)$ that we use to quantify the alignment solution vector are listed in Table 2 and for the illustration of derivation, we mainly use the squared L_2 norm utility function.

Table 2: Choices of utility functions $f(\cdot)$ w.r.t. the solution vector x of network alignment. Here, $\log(\mathbf{x})$ represents element-wise logarithmic function[1] and $\|\cdot\|_1$ is L_1 norm.

Description	Function $f(\cdot)$
Squared L_2 norm	$f(\mathbf{x}) = \|\mathbf{x}\|_2^2$
Entropy	$f(\mathbf{x}) = \|\mathbf{x} \odot \log(\mathbf{x})\|_1$

In this paper, we propose to leverage influence functions [12] w.r.t. the alignment solution vector $\mathbf{x}$ to quantify the informativeness of queries. Let us start with two definitions of the influence of different query types on the network alignment result.

[1] We define $0 \log 0 = 0$.

Definition 1. *Node Pair Query Influence on Attributed Network Alignment.* Given a selected utility function over the alignment solution vector, i.e., $f(\mathbf{x})$, the influence of a node pair query, denoted as q_{ij} where node i is in $\mathcal{G}_t$ and node j is in $\mathcal{G}_s$, w.r.t. $f(\mathbf{x})$ is defined as the derivative of $f(\mathbf{x})$ with respect to the preference for aligning this pair of nodes (i.e., $\mathbf{H}(i,j)$). The influence of the node pair query is formally defined as $\mathcal{I}(q_{ij}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{H}(i,j)}$.

Definition 2. *Node Query Influence on Attributed Network Alignment.* For a node query, denoted as q_j where node j is in the source network $\mathcal{G}_s$, the influence is defined as the aggregation of the influences of node pairs between node j in $\mathcal{G}_s$ and all the nodes in $\mathcal{G}_t$, i.e., $\mathcal{I}(q_j) = \sum_{i=1}^{n_t} \mathcal{I}(q_{ij})$.

Next, we present the details on how to calculate the influence of the two types of queries w.r.t. the alignment result given a utility function $f(\cdot)$.

A - Node-Pair query influence. We first compute the node pair query influence on network alignment in Lemma 1.

Lemma 1. (Node Pair Query Influence on Attributed Network Alignment.) For a selected utility function over the alignment solution vector, i.e., $f(\mathbf{x})$, the influence of a specific node pair (i,j) w.r.t. $f(\mathbf{x})$ can be computed as follows,

$$\mathcal{I}(q_{ij}) = \begin{cases} 2\mathbf{P}(k,:)\mathbf{x} & \text{Squared } L_2 \text{ norm} \\ -\mathbf{P}(k,:)(\log(\mathbf{x}) + \mathbf{1}) & \text{Entropy} \end{cases} \tag{3}$$

where $\mathbf{P} = (1-\alpha)(\mathbf{I} - \alpha\widetilde{\mathbf{W}})^{-1}$ and $\mathbf{P}$ is symmetric, $\mathbf{I} \in \mathbb{R}^{n_s n_t \times n_s n_t}$ is an identity matrix, $k = (j-1)\cdot n_t + i$ is the new index of $\mathbf{H}(i,j)$ after we vectorize $\mathbf{H}$ in the column order, the log operator is applied element-wisely and $\mathbf{1}$ is an all-ones vector.

Proof. (i). We first prove the case of using squared L_2 norm as the utility function, i.e., $f(\mathbf{x}) = \|\mathbf{x}\|_2^2$. According to Eq. (1), the closed-form solution of network alignment is given as,

$$\mathbf{x} = (1-\alpha)(\mathbf{I} - \alpha\widetilde{\mathbf{W}})^{-1}\mathbf{h} \tag{4}$$

Following Definition 1, we apply chain rule [19, Page 15] and take the partial derivative of $f(\mathbf{x})$ w.r.t. the specific node pair (i.e., $\mathbf{H}(i,j)$),

$$\mathcal{I}(q_{ij}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{H}(i,j)} = [\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}]' \frac{\partial \mathbf{x}}{\partial \mathbf{H}(i,j)} \tag{5}$$

Here, for the first derivative, it is easy to obtain $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = 2\mathbf{x}$, we also vectorize the preference matrix $\mathbf{H}$ and denote the new index of $\mathbf{H}(i,j)$ in $\mathbf{h}$ as $k = (j-1)\cdot n_t + i$ where n_t is the size of $\mathcal{G}_t$. By letting $\mathbf{P} = (1-\alpha)(\mathbf{I} - \alpha\widetilde{\mathbf{W}})^{-1}$, for the second partial derivative, we further have,

$$\frac{\partial \mathbf{x}}{\partial \mathbf{h}_k} = \frac{\partial (\mathbf{P}\mathbf{h})}{\partial \mathbf{h}_k} = \mathbf{P}\mathbf{e}_k = \mathbf{P}(:,k) \tag{6}$$

where $\mathbf{e}_k$ is a single-entry vector of length $n_s \times n_t$ with 1 at the k^{th} position and the second partial derivative returns the k^{th} column of matrix $\mathbf{P}$. Since $\mathbf{P}$ is symmetric, by combining everything, we obtain the influence of the node pair (i,j) w.r.t. the squared L_2 norm of network alignment solution vector as follows,

$$\mathcal{I}(q_{ij}) = 2\mathbf{P}(:,k)'\mathbf{x} = 2\mathbf{P}(k,:)\mathbf{x} \tag{7}$$

(ii). For the case of entropy utility function, it is obvious that the entropy of solution matrix $\mathbf{X}$ is the summation of the entropy of each column in $\mathbf{X}$, i.e.,

$$f(\mathbf{X}) = \sum_{s=1}^{n_s} f(\mathbf{X}(:,s)) \tag{8}$$

We start from computing the partial derivative of the entropy of a column in $\mathbf{X}$ (e.g., the s^{th} column $\mathbf{X}(:,s)$) w.r.t. $\mathbf{H}(i,j)$, according to the definition in Table 2, we have

$$\begin{aligned}\frac{\partial f(\mathbf{X}(:,s))}{\partial \mathbf{H}(i,j)} &= -\frac{\partial \sum_{t=1}^{n_t} \mathbf{X}(t,s)\log \mathbf{X}(t,s)}{\partial \mathbf{H}(i,j)} = -\sum_{t=1}^{n_t} \frac{\partial[\mathbf{X}(t,s)\log \mathbf{X}(t,s)]}{\partial \mathbf{H}(i,j)} \\ &= -\sum_{t=1}^{n_t}[\frac{\partial \mathbf{X}(t,s)}{\partial \mathbf{H}(i,j)}\log \mathbf{X}(t,s) + \frac{\partial \mathbf{X}(t,s)}{\partial \mathbf{H}(i,j)}] \\ &= -\sum_{t=1}^{n_t}\frac{\partial \mathbf{X}(t,s)}{\partial \mathbf{H}(i,j)}(\log \mathbf{X}(t,s)+1)\end{aligned} \tag{9}$$

Similarly, we denote $l = (s-1)\cdot n_t + t$, $k = (j-1)\cdot n_t + i$ as the new indexes of the corresponding entries in the vectorized $\mathbf{X}$ and $\mathbf{H}$, respectively. The partial derivative in Eq. (9) is,

$$\frac{\partial \mathbf{X}(t,s)}{\partial \mathbf{H}(i,j)} = \frac{\partial \mathbf{x}_l}{\partial \mathbf{h}_k} = \frac{\partial \mathbf{P}(l,:)\mathbf{h}}{\partial \mathbf{h}_k} = \mathbf{P}(l,:)\mathbf{e}_k = \mathbf{P}(l,k) \tag{10}$$

We further have,

$$\frac{\partial f(\mathbf{X}(:,s))}{\partial \mathbf{H}(i,j)} = -\sum_{l=(s-1)\cdot n_t+1}^{sn_t} \mathbf{P}(l,k)(\log \mathbf{x}_l + 1) \tag{11}$$

According to Eq. (8), by putting everything together, we compute the node pair query influence w.r.t. the entropy of the alignment solution vector as follows,

$$\begin{aligned}\mathcal{I}(q_{ij}) &= -\sum_{s=1}^{n_s}\sum_{l=(s-1)\cdot n_t+1}^{sn_t} \mathbf{P}(l,k)(\log \mathbf{x}_l + 1) \\ &= -\mathbf{P}(k,:)(\log(\mathbf{x}) + \mathbf{1})\end{aligned} \tag{12}$$

which completes the proof. □

B - Node query influence. According to the node pair query influence in Lemma 1 and Definition 2, we can compute the node query influence in a straight-forward way, which is summarized in the following lemma.

Lemma 2. (Node Query Influence to Attributed Network Alignment.) Given a utility function over the alignment solution vector, i.e., $f(\mathbf{x})$, the influence of a specified node query (e.g., node j in $\mathcal{G}_s$), denoted as q_j, w.r.t. $f(\mathbf{x})$ can be computed as follows,

$$\mathcal{I}(q_j) = \begin{cases} 2\|\mathbf{P}(k_1:k_2,:)\mathbf{x}\|_1 & \text{Squared } L_2 \text{ norm} \\ \|\mathbf{P}(k_1:k_2,:)(\log(\mathbf{x})+\mathbf{1})\|_1 & \text{Entropy}\end{cases} \tag{13}$$

where $k_1 = (j-1)\cdot n_t + 1$, $k_2 = j\cdot n_t$, $\mathbf{P} = (1-\alpha)(\mathbf{I}-\alpha\widetilde{\mathbf{W}})^{-1}$, $\mathbf{1}$ is an all-ones vector and $\|\cdot\|_1$ represents L_1 norm.

PROOF. (i). For the case of using squared L_2 norm as the utility function, according to Definition 2, the node query influence is the summation of influences of all node pairs between node j in $\mathcal{G}_s$ and all the nodes in $\mathcal{G}_t$, and based on Eq. (7) for node-pair query influence, we have,

$$\mathcal{I}(q_j) = \sum_{i=1}^{n_t}\mathcal{I}(q_{ij}) = 2\sum_{k=(j-1)n_t+1}^{jn_t} \mathbf{P}(k,:)\mathbf{x} \tag{14}$$

which indicates that by selecting the sub-matrix of $\mathbf{P}$ from the $((j-1)n_t+1)^{\text{th}}$ row to the $(jn_t)^{\text{th}}$ row (i.e., $\mathbf{P}(k_1:k_2,:)$) and aggregating the entries of the resulting vector (i.e., applying L_1 norm on $\mathbf{P}(k_1:k_2,:)\mathbf{x}$), we obtain the equation for computing the node query influence as follows,

$$\mathcal{I}(q_j) = 2\|\mathbf{P}(k_1:k_2,:)\mathbf{x}\|_1 \tag{15}$$

(ii). The proof for the case of entropy is similar as (i) by selecting the same sub-matrix of $\mathbf{P}$ of rows at the same indexes, followed by applying L_1 norm on the resulting vector (i.e., $\mathbf{P}(k_1:k_2,:)\mathbf{x}$). Therefore, the node query influence w.r.t. the entropy is,

$$\mathcal{I}(q_j) = \|\mathbf{P}(k_1:k_2,:)(\log(\mathbf{x})+\mathbf{1})\|_1 \tag{16}$$

which completes the proof. □

Eq. (13) requires explicitly calculating the matrix $\mathbf{P} = (1-\alpha)(\mathbf{I}-\alpha\widetilde{\mathbf{W}})^{-1}$, which is computationally expensive. To address this problem, we first denote the new vector $\mathbf{y} = \mathbf{P}\mathbf{x}$ (i.e., the case of squared L_2 norm as the utility function) where $\mathbf{x}$ is the alignment solution vector. To solve for $\mathbf{y}$, we have

$$\begin{aligned}&\mathbf{y} = (1-\alpha)(\mathbf{I}-\alpha\widetilde{\mathbf{W}})^{-1}\mathbf{x} \\ \Leftrightarrow\ &\mathbf{y} = \alpha\widetilde{\mathbf{W}}\mathbf{y} + (1-\alpha)\mathbf{x}\end{aligned} \tag{17}$$

which implies that $\mathbf{y}$ can be computed by the iterative power method and therefore, the direct computation of the inverse matrix can be avoided. For the case of entropy, we have the following,

$$\mathbf{y} = \alpha\widetilde{\mathbf{W}}\mathbf{y} + (\alpha-1)(\log(\mathbf{x})+\mathbf{1}) \tag{18}$$

We further observe that the influence of query for node j can be calculated by the following equivalent equation,

$$\mathcal{I}(q_j) = \|\mathbf{y}_{k_1:k_2}\|_1 \tag{19}$$

where $k_1 = (j-1)\cdot n_t + 1$, $k_2 = j\cdot n_t$, which means that we can calculate $\mathcal{I}(q_j)$ by aggregating the corresponding entries in $\mathbf{y}$.

4.2 Proposed Algorithm

A - A generic algorithm for active attributed network alignment. Based on Lemma 1 and Lemma 2. We propose Algorithm 1 to select the most informative nodes to query so as to improve the performance of network alignment algorithms. The key idea of the proposed algorithm is that we iteratively select the nodes with the largest influence calculated by our proposed quantitative method in Eq. (19) (Step-7, Step-10), query the oracle for their correct alignment (Step-13), update the alignment preference matrix, i.e., $\mathbf{H}$ in Eq. (1) accordingly (Step-14), recompute the alignment results (Step-15) and the influence of the remaining candidate query nodes. (How the query results are utilized to update the network alignment will be presented in the experiment setting in Section 5.1.) In Algorithm 1, the alignment preference vector is initialized as uniform unit vector, i.e., $\mathbf{h} = \mathbf{1}\times\frac{1}{n_t\times n_s}$, which means that we do not have any prior knowledge of the alignment result. To update $\mathbf{H}$, for example, if the query result gives that node j in $\mathcal{G}_s$ is aligned to node i in $\mathcal{G}_t$, we set $\mathbf{H}(i,j) = 1$ while keeping the remaining values (i.e., $\mathbf{H}(i,k)|_{k\neq j}$ and $\mathbf{H}(k,j)|_{k\neq i}$) unchanged.

Note that Algorithm 1 provides a family of query strategies for active network alignment based on the selected utility function $f(\cdot)$. We use different suffixes to differentiate utility functions, i.e., ATTENT-L2 and ATTENT-Entropy for using squared L_2 norm and entropy, respectively.

Algorithm 1 A generic algorithm for active network alignment

Input: (1) Two attributed networks $\mathcal{G}_s$, $\mathcal{G}_t$; (2) an integer query budget k; (3) a utility function $f(\cdot)$; (4) a network alignment algorithm $\mathcal{A}$; (5) the size b for batch query; (6) alignment preference matrix $\mathbf{H}$
Output: a set of k nodes in $\mathcal{G}_s$ for query, Q, and the alignment result $\mathcal{M}$ between two networks $\mathcal{G}_s$ and $\mathcal{G}_t$.

Initialize the query node set $Q = \emptyset$, query pool $J = V_s$;
Compute the current network alignment result $\mathcal{M}$ using the specified alignment algorithm $\mathcal{A}$;
Remove the nodes that are correctly aligned based on $\mathcal{M}$ from the query pool I;
while $|Q| < k$ **do**
  Compute alignment solution vector $\mathbf{x}$ using Eq. (2);
  Compute $\mathbf{y}$ using Eq. (17) or (18);
  Compute influence for all nodes in I using Eq. (19);
  Initialize $Q_s = \emptyset$;
  while $|Q_s| < b$ **do**
    Add node $v^\star = \text{argmax}_{j \in J}\, \mathcal{I}(q_j)$ to Q_s;
    Update $J \leftarrow J \setminus \{v^\star\}$;
  end while
  Query for the correct alignment of the nodes in Q_s;
  Update the alignment preference matrix $\mathbf{H}$;
  Re-compute the network alignment result $\mathcal{M}$ using $\mathcal{A}$;
  Update $Q = Q \cup Q_s$;
  Update the query pool J by removing newly aligned nodes according to $\mathcal{M}$;
end while
return Q and $\mathcal{M}$;

B - Speed-up Computation. The main computational bottleneck of the proposed query strategy in Algorithm 1 is that in each iteration, we need to re-compute the alignment solution vector $\mathbf{x}$ in Eq. (2) using a fixed-point method and the influence of each remaining candidate query using Eq. (19). We first analyze the time and space complexity of the proposed query method Attent-Entropy/L2 with straightforward implementation, as summarized in the following lemma.

Lemma 3. (Time and space complexity of Attent-Entropy/L2.) The time complexity of Attent-Entropy/L2 is $O(\frac{k}{b}(Lmnt_{max} + LKn^2 + m^2 t_{max}))$ and the space complexity is $O(n^2 + m^2)$. where t_{max} is the maximum number of iterations, n, m are the number of nodes and edges of the network respectively, k is the query budget and b is the batch size.

Proof. There are two main steps in Attent-Entropy/L2, including iteratively (1) computing alignment solution vector $\mathbf{x}$, and (2) computing vector $\mathbf{y}$ in Eq. (17) or Eq. (18) for query influence. According to [31], it takes $O(Lmnt_{max} + LKn^2)$ time and $O(n^2)$ space to compute the alignment vector. To compute $\mathbf{y}$, the Kronecker product $\mathbf{A}_s \otimes \mathbf{A}_t$ takes $O(m^2)$ time and space, which implies computing $\widetilde{\mathbf{W}}$ takes $O(m^2)$ time and space, therefore the power method requires $O(m^2 t_{max})$ in time and $O(n^2 + m^2)$ in space. Combining everything together, we have the time and space complexity for Attent-Entropy/L2 are $O(\frac{k}{b}(Lmnt_{max} + LKn^2 + m^2 t_{max}))$ and $O(n^2 + m^2)$, respectively, which completes the proof. □

In order to further improve the efficiency of the proposed methods, we propose a fast and exact solution to speed up and scale up the computation of the query influence. Some recent progress suggests that Sylvester equation in the form of Eq. (1) can be solved in linear time *without* quality loss [10], as shown in the following proposition:

Proposition 1. By implicit Kronecker Krylov subspace method [10], Eq. (1) for attributed network alignment can be solved in $O(rm + r^2 Kn)$ time, where n, m are the numbers of nodes and edges in the input networks, respectively, r is the Krylov subspace size and K is the number of node attributes.

Based on Proposition 1, we propose Attent-Entropy/L2-Fast to speed up and scale up the computation of query influence. The key idea behind our proposed fast solution is two-fold. First, at each iteration, we first leverage Kronecker Krylov subspace method to solve for the current alignment solution, i.e., $\mathbf{x}$ in the linear system Eq. (1) of the updated alignment preference vector (i.e., $\mathbf{h}$) and get the current alignment solution $\mathbf{x}$. Second, in Attent-L2-Fast, we replace the alignment preference vector with the obtained $\mathbf{x}$, apply the Kronecker Krylov method to solve the new Eq. (1) and obtain $\mathbf{y}$ in Eq. (17). Note that in Attent-Entropy-Fast, we replace $\mathbf{h}$ with $-(\log(\mathbf{x}) + \mathbf{1})$. The influence of query can then be computed using Eq. (19).

The time and space complexity of Attent-Entropy/L2-Fast is given in the following lemma,

Lemma 4. (Time and space complexity of Attent-Entropy/L2-Fast.) The time complexity of Attent-Entropy/L2-Fast is $O(\frac{k}{b}(rm + r^2 Kn))$ and the space complexity is $O(rm + rKn)$ where n, m are the number of nodes and edges of the network respectively, k is the query budget, b is batch size and r is the Krylov subspace size.

Proof. It directly follows Proposition 1 and Lemma 3. Omitted for brevity. □

5 EXPERIMENT

In this section, we perform experiments to evaluate the proposed query strategies for active attributed network alignment to answer the following two research questions:

- **RQ1 Effectiveness.** How accurate are the proposed algorithms in querying the most informative nodes to improve the network alignment accuracy? How attribute information contributes to the query strategy?
- **RQ2 Efficiency.** How fast and scalable are the proposed algorithms of influence based query strategy?

5.1 Experimental Setup

A - Datasets We use 6 real-world datasets, which are publicly available. The detailed descriptions of these datasets are as follows,

- **ACM Citation** was collected in 2016 from 2,381,688 papers and is a co-authorship network where nodes represent authors, and there is an edge between two authors if they have published a paper together. Numerical node attributes represent the ratio of papers that authors have published in 17 venues of four areas of computer science research, including data mining, machine learning, database and information retrieval [30].

Table 3: Statistics of datasets.

Dataset	Avg. #nodes	Avg. #edges	Node Attribute	Edge Attribute
ACM Citation	9,872	79,122	17	17
DBLP Citation	9,916	89,616	17	17
Douban	3,906	8,164	538	2
Lastfm	15,436	16,319	3	3
Flickr	12,974	16,149	3	3
AMiner	1,274,360	8,273,167	3	1

- **DBLP Citation** was collected from 3,272,991 papers in 2016 and is a co-authorship network where the node attributes represent 17 research conferences in four areas as in *ACM Citation* [30].
- **Douban** contains 50k users and 5M edges. Information such as location and offline event participation are included in user's profiles. The edge attributes represent whether two users are contacts or friends [36].
- **Lastfm** contains 136,420 users and 1,685,524 following relationships. For each user's profile, some profile details are recorded such as age, gender, locations and so on [35].
- **Flickr** is a popular photo-sharing network that allows users to upload and share photos with others. The Flickr dataset consists of individual users and friendship relations. The profile of users includes gender, hometown, occupation and so on [35].
- **AMiner** is an academic social network. Undirected edges represent co-authorship and the node attribute vectors represents the number of published papers [35].

B - Experiment Setting. We evaluate the effectiveness of our proposed algorithms by measuring the alignment accuracy of the remaining nodes in the source network $\mathcal{G}_s$ which have not been queried. The subsets of the aforementioned datasets are used to construct various alignment scenarios, whose statistics are summarized in Table 3.

For the regularization parameter α in Eq. (1), we set α to be 0.6, and we find that the for the value of α between 0.2 to 0.8, the results do not show a large difference. We perform the evaluations in the following four alignment scenarios, including (1) *ACM-ACM Co-authorship*, we extract three subgraphs from the constructed *ACM Citation* co-authorship network and the subgraphs contain an average of 5,502 nodes and 23,406 edges. For each subgraph, we then add noisy edge weights to the adjacency matrix, randomly permute the modified subgraph and treat it as the second network. We set the query budget $k = 200$ and select $b = 20$ nodes to query at each iteration and report the alignment accuracy as the final results; (2) *ACM-DBLP Co-authorship*, we extract one pair of subgraphs from *ACM* co-authorship network and *DBLP* co-authorship network, respectively. For subgraph from *ACM Citation*, it contains 9,817 nodes and 36,619 edges, the number of nodes and edges for subgraph from *DBLP Citation* is 9,886 and 36,260, respectively. We set $k = 400$ and $b = 40$ and report the alignment accuracy; (3) *Flickr-Lastfm*, we extract subgraphs from *Flickr* and *Lastfm*, respectively, given the partial ground-truth of 452 aligned pairs of nodes [35], we set the query budget $k = 200$ and the batch size $b = 20$ and report the alignment accuracy as the results; (4) *Douban offline-Douban online*, the partial ground-truth contains 1,118 nodes of correct matching, we set the query budge k and the batch size b to be 100 and 10, respectively; (5) *AMiner-AMiner*, we use this scenario for scalability evaluations because it contains the largest networks, and we set $k = 100, b = 10$.

To demonstrate the applicability of the proposed influence function based query strategy, we select three network alignment algorithms to evaluate the effectiveness, including (1) the optimization based attributed network alignment algorithm *FINAL* [30]; (2) *NetAlign* algorithm based on belief propagation [2], and (3) *IsoRank* algorithm [25]. For *FINAL* and *IsoRank* algorithms, after we obtain the query results from the oracle, the alignment preference matrix $\mathbf{H}$ will be updated accordingly. For example, if node i in the target network $\mathcal{G}_t$ matches the query node j in the source network $\mathcal{G}_s$, then we can set $\mathbf{H}(i, j)$ to be a scalar which satisfies $\mathbf{H}(i, j) \gg \frac{1}{n_s \cdot n_t}$ (e.g., 1), which ensures that in the alignment solution matrix $\mathbf{X}$, we get $i = \arg\max_k \mathbf{X}(k, j)$. For *NetAlign* algorithm, the weights of edges in the bipartite graph represent the cross-network node similarity and therefore can be modified according to the query results, and please refer to [2] for more details.

C - Comparison Methods. We compare our proposed algorithms with several baseline methods and the detailed descriptions are summarized as follows,

- *TopMatchings* and *GibbsMatchings*, are two matching-based query strategies for active network alignment [17]. The key idea is that given l possible matchings that are sampled from *TopMatchings* or *GibbsMatchings*, if one node in the source network is aligned to different nodes, then this node is considered to be uncertain therefore it will be a potential candidate to query. Given the set of candidate matchings for node i, the certainty is defined as marginal distribution, which can be computed as the fraction of matchings where node i is correctly aligned to node j, i.e., $\text{certainty}(q_i) = \max_{j \in P_i} \Pr(\mathcal{M}(i) = j|\mathcal{M})$, and this query strategy selects node i with the least certainty, i.e., $\arg\min_{i \in V_s} \text{certainty}(q_i)$.
- *Entropy*, the solution matrix $\mathbf{X}$ in Eq. (1) represents the cross network node similarity. The intuition behind this query strategy is that if one row/column of $\mathbf{X}$ is close to uniform distribution, then the uncertainty of aligning the corresponding node will be large, i.e., the entropy of the similarity distribution is large. We then query the node with the largest entropy, $q_i = \arg\max_{i \in V_s} \sum_{j \in V_t} -\mathbf{X}(j, i) \log \mathbf{X}(j, i)$
- *Margin*, follows the idea in active learning that data samples with small differences between the two most probable labels are likely to be good query candidates [21]. Given the alignment solution matrix $\mathbf{X}$, we define the informativeness of a node i as $\mathcal{I}(q_i) = \mathbf{X}(p, i) - \mathbf{X}(q, i)$ where $\mathbf{X}(p, i), \mathbf{X}(q, i)$ are the two largest values in $\mathbf{X}(:, i)$. The smaller difference between $\mathbf{X}(p, i)$ and $\mathbf{X}(q, i)$, the more uncertain node i will be.
- *Least confident*, follows the idea of a common query strategy in active learning that a data sample whose prediction is the least confident is likely to be a good candidate for query. The informativeness of a query q_i is defined as, $\mathcal{I}(q_i) = 1 - \max_j \mathbf{X}(j, i)$. The larger $\mathcal{I}(q_i)$ is, the more uncertain node i will be. This query strategy selects the nodes with the largest $\mathcal{I}(q_i)$.
- *Betweenness*, selects the candidate nodes for query which have the largest betweenness centrality in the source network $\mathcal{G}_s$ [16].
- *Random*, selects candidate queries from the source network $\mathcal{G}_s$ randomly.

When evaluating the comparison methods, including *Entropy*, *Margin* and *Least confident*, we apply *FINAL* [30] algorithm to compute the alignment solution matrix **X** where the alignment preference utilizes node degree similarity or node attribute similarity instead of the initialization we use in evaluate Attent-Entropy/L2.

D - Machine Configuration and Repeatability. The experiments are performed on an Ubuntu 18.04 desktop with Intel Core i7-9800X at 3.80GHz and 64GB RAM, and are implemented in Matlab. All datasets are publicly available.

5.2 Effectiveness Results

A - Alignment accuracy. We first compare the proposed influence based query strategies (Attent-L2 and Attent-Entropy) in alignment accuracy with other query methods on three different network alignment algorithms, including *FINAL*, *NetAlign* and *IsoRank*. The results are summarized in Figures 2, 3 and 4, respectively. We can observe that the proposed influence based query strategies, Attent-L2 and Attent-Entropy (brown and red lines with the marker of circle and diamond, respectively) achieve the best performance over all other comparison query strategies in all four alignment scenarios. For example, in the alignment scenario of *ACM Co-authorship* using three network alignment algorithms, the proposed query strategy achieves 0.59%, 3.85%, 3.63% higher alignment accuracy than the best competitor strategy after the query is complete, i.e., Attent-L2 vs. *GibbsMatchings* in Figure 2a, 3a and 4a, respectively. In aligning the *Flickr-Lastfm* networks, Attent-L2 again outperforms *GibbsMatchings* in accuracy by 4.15%, 2.37%, and 3.09% for three alignment algorithms (i.e., Attent-L2 vs. *GibbsMatchings* in Figure 2c, 3c and 4c). For the other proposed query strategy, i.e., Attent-Entropy, we have the following two observations. First, it achieves better alignment accuracy compared with baseline query methods. For example, in aligning *Douban* networks, after 100 queries, it achieves 1.95%, 1.74% and 1.26% higher accuracy than *TopMatchings* on three alignment algorithms, respectively. Second, it has a comparable performance with Attent-L2 in alignment accuracy. For example, in Figure 3b of aligning *ACM-DBLP* networks using the *NetAlign* algorithm, the difference of accuracy between the two proposed query strategies is less than 0.1% after all queries are finished. As shown in Figure 2d, Attent-L2 outperforms Attent-Entropy in accuracy by 1.84%. On the other hand, Attent-Entropy achieves 2.87% better alignment accuracy than Attent-L2 on *ACM-DBLP* dataset in Figure 4b.

B - Parameter analysis. There are two key parameters in the proposed Attent-Entropy/L2 strategy for active network alignment, including (1) the query budget k, and (2) the batch size b. The impact of the query budget k is presented in Figures 2-4, which show that the larger the budget k, the higher the alignment accuracy. We perform a parametric study on the batch size b in Figure 5a. In our study, we test the proposed Attent-L2 on *ACM-DBLP* networks with a fixed query budget $k = 400$ but different batch size b. We have the following two observations. First, the proposed Attent-L2 achieves the highest alignment accuracy when the batch size $b = 20$ (i.e., the blue line with square marker). Second, as the batch size increases, the performance slightly decreases to a small extent. For example, the alignment accuracy for $b = 40$ (i.e., the orange line) is 1.12% less than that for $b = 20$ after 400 queries, and when the batch size $b = 100$ (i.e., the purple line), the alignment accuracy is 3.73% less than the highest accuracy (i.e., the blue line). Therefore the batch size b has a relatively small impact on the performance of the proposed algorithm.

C - Ablation study. To evaluate to what extent node/edge attributes contribute to the performance gain of the proposed query strategy, we perform experiments on the proposed Attent-L2 with and without attributes (i.e., **N** and **E**). Table 4 summarizes the performance of three query strategies w.r.t. alignment accuracy vs. # queries. We denote Attent-L2 with attributes, Attent-L2 without attributes and *GibbsMatchings* as $\mathrm{S_A}$, $\mathrm{S_A^\star}$ and $\mathrm{S_G}$, respectively. We have the following observations. First, even without attributes, our proposed Attent-L2 ($\mathrm{S_A^\star}$) still outperforms *GibbsMatchings* ($\mathrm{S_G}$) — the best performing baseline method. Second, by leveraging node and edge attribute information, the alignment accuracy of Attent-L2 ($\mathrm{S_A}$) is further improved, which corroborates the effectiveness of attributes in our proposed methods.

Table 4: Ablation study on alignment accuracy (%) vs. #queries on *ACM-DBLP* and *Douban* networks using *FINAL* algorithm. ($\mathrm{S_A}$: Attent-L2 with attributes, $\mathrm{S_A^\star}$: Attent-L2 without attribute, $\mathrm{S_G}$: GibbsMatchings)

	ACM-DBLP				Douban		
k	$\mathrm{S_A}$	$\mathrm{S_A^\star}$	$\mathrm{S_G}$	k	$\mathrm{S_A}$	$\mathrm{S_A^\star}$	$\mathrm{S_G}$
40	47.54	46.87	46.09	10	22.64	22.37	22.19
80	52.45	50.23	47.92	20	24.86	23.75	23.07
120	53.92	52.78	51.41	30	25.28	25.16	25.05
160	55.95	54.73	53.90	40	26.41	26.22	25.21
200	57.65	55.91	54.46	50	27.01	26.49	26.21

5.3 Efficiency Results

Scalability. The scalability results of the proposed methods on *AMiner* dataset are presented in Figure 5b. We can see that the proposed fast solutions (Attent-L2-Fast and Attent-Entropy-Fast) scale linearly w.r.t. the size of the input networks, which is consistent with Lemma 4. The results for Attent-Entropy and Attent-L2 are also consistent with Lemma 3 of the complexity analysis.

Accuracy-speed trade-off. The query quality vs. running time trade-off of all methods including proposed and comparison ones on four datasets using *FINAL* algorithm is shown in Figure 6. We observe that the proposed query strategy including Attent-Entropy/L2 and their fast solutions, i.e., Attent-Entropy/L2-Fast, achieve a good balance between the alignment accuracy and running time. For example, in Figure 6c, the total query time of Attent-L2 and Attent-Entropy (upper left region circled by the dashed line) is less than 1/4 of that of *TopMatchings* (672.2*s* vs. 3371.4*s*), while outperforming it in alignment accuracy. The proposed fast solutions (Attent-Entropy-Fast and Attent-L2-Fast) significantly reduce the query time and are 72× faster than *TopMatching* (46.3*s* vs. 3371.4*s*) , and are also 15× faster than Attent-Entropy/L2 with the same alignment accuracy. Among the baselines, *GibbsMatchings* is most competitive, with a similar query time as the proposed Attent-Entropy/L2-Fast. The accuracy of *GibbsMatchings* is on a

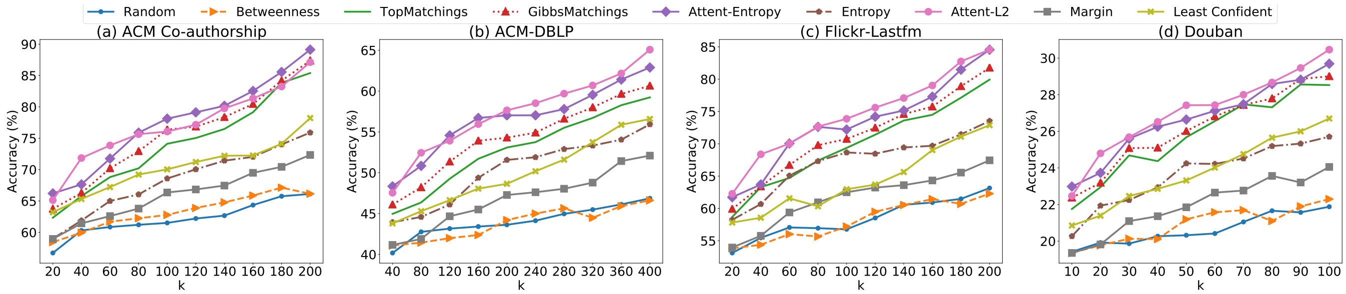


Figure 2: Alignment accuracy vs. # queries in four alignment scenarios using *FINAL* algorithm. Tthe top two solid lines represent the proposed Attent-Entropy and Attent-L2 respectively. Best viewed in color.

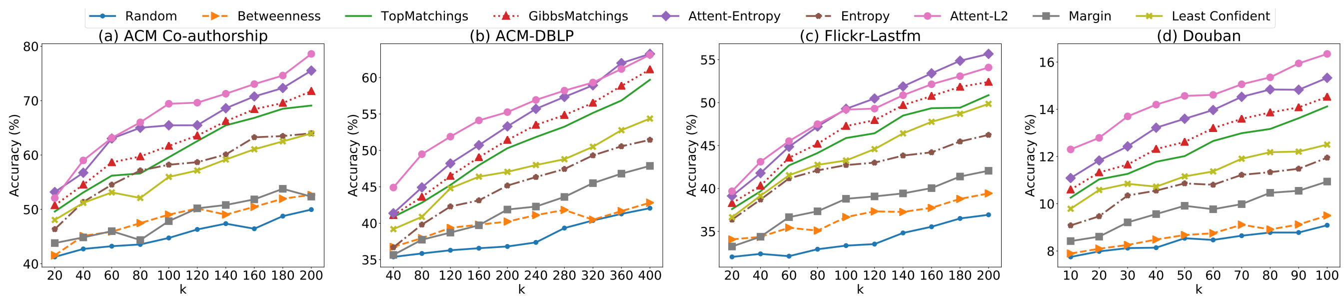


Figure 3: Alignment accuracy vs. # queries in four alignment scenarios using *NetAlign* algorithm. The top two solid lines represent the proposed Attent-Entropy and Attent-L2 respectively. Best viewed in color.

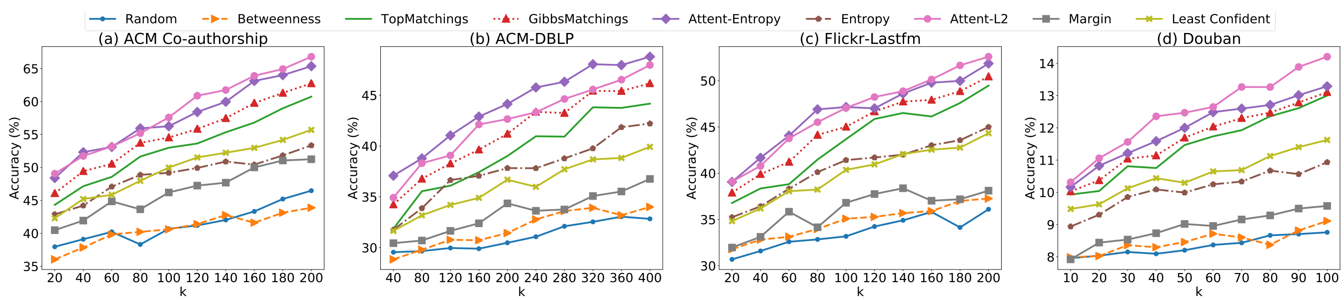
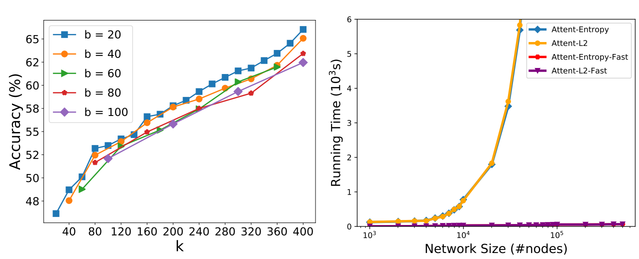


Figure 4: Alignment accuracy vs. # queries in four alignment scenarios using *IsoRank* algorithm. The top two solid lines represent the proposed Attent-Entropy and Attent-L2 respectively. Best viewed in color.



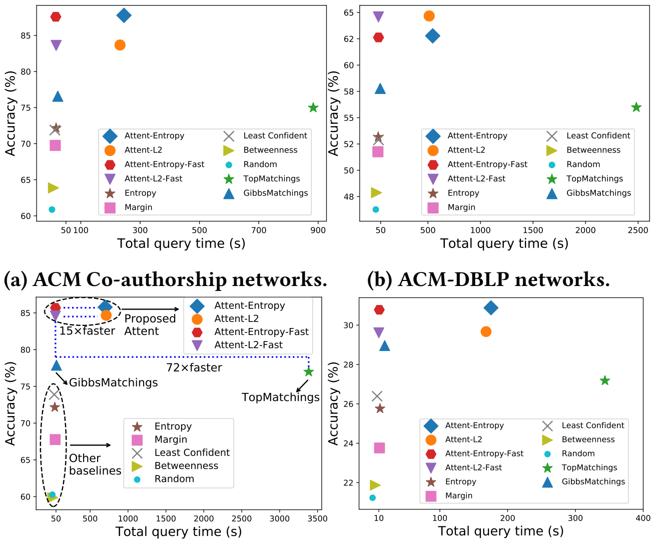
(a) Parameter analysis **(b) Scalability**

Figure 5: (a) Parameter analysis on *ACM-DBLP* dataset; (b) running time vs. number of nodes on *AMiner* dataset. Best viewed in color.

par with *TopMatchings*, both of which are lower than the proposed Attent. Other baseline methods (e.g., lower left region circled by the dashed line in Figure 6c), although they are fast (x-axis), they are much less effective in improving network alignment accuracy (y-axis).

6 CONCLUSION

In this paper, we study the problem of active attributed network alignment. The key idea is to characterize the change of alignment results w.r.t. query nodes and select the most informative ones to query so as to maximally improve the alignment accuracy. We propose a family of algorithms (Attent) capable of measuring the influence of query on the alignment results, along with the fast solutions of a linear complexity in both time and space. We demonstrate the efficacy of the proposed query strategy through extensive empirical evaluations on real-world datasets. The proposed query strategy for active network alignment is applicable to a variety of network alignment algorithms. Future work includes generalizing the current active learning approach in other multi-network mining tasks as well as developing reinforcement learning-based methods for active network mining.



(a) ACM Co-authorship networks. **(b) ACM-DBLP networks.**

(c) Flickr-Lastfm networks. **(d) Douban networks.**

Figure 6: Balance between alignment accuracy and total query time. Best viewed in color.

ACKNOWLEDGEMENT

This work is supported by National Science Foundation under grant No. 2003924, by the NSF Program on Fairness in AI in collaboration with Amazon under award No. 1939725, and IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network. The content of the information in this document does not necessarily reflect the position or the policy of the Government or Amazon, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] Sergey Bartunov, Anton Korshunov, Seung-Taek Park, Wonho Ryu, and Hyungdong Lee. 2012. Joint link-attribute user identity resolution in online social networks.
[2] Mohsen Bayati, Margot Gerritsen, David F Gleich, Amin Saberi, and Ying Wang. [n. d.]. Algorithms for large, sparse network alignment problems. In *2009 IEEE ICDM*.
[3] Mustafa Bilgic, Lilyana Mihalkova, and Lise Getoor. 2010. Active learning for networked data. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 79–86.
[4] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2017. Active learning for graph embedding. *arXiv preprint arXiv:1705.05085* (2017).
[5] Xiaokai Chu, Xinxin Fan, Di Yao, Zhihua Zhu, Jianhui Huang, and Jingping Bi. 2019. Cross-Network Embedding for Multi-Network Alignment. In *The World Wide Web Conference, WWW 2019*. ACM.
[6] Connor Clark and Jugal Kalita. 2014. A comparison of algorithms for the pairwise alignment of biological networks. *Bioinformatics* (2014), 2351–2359.
[7] Xavier Cortés and Francesc Serratosa. 2013. Active-learning query strategies applied to select a graph node given a graph labelling. In *International Workshop on Graph-Based Representations in Pattern Recognition*. Springer, 61–70.
[8] Kaize Ding, Jundong Li, and Huan Liu. 2019. Interactive anomaly detection on attributed networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 357–365.
[9] Kaize Ding, Jianling Wang, Jundong Li, Kai Shu, Chenghao Liu, and Huan Liu. 2020. Graph prototypical networks for few-shot learning on attributed networks. In *Proceedings of the 29th ACM CIKM*. 295–304.
[10] Boxin Du and Hanghang Tong. 2018. FASTEN: Fast Sylvester equation solver for graph mining. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1339–1347.
[11] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. Regal: Representation learning-based graph alignment. In *CIKM*.
[12] Pang Wei Koh and Percy Liang. 2017. Understanding Black-box Predictions via Influence Functions. In *Proceedings of the 34th ICML*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 1885–1894.
[13] Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. 2017. Learning Active Learning from Data. In *Advances in Neural Information Processing Systems*.
[14] Yayong Li, Jie Yin, and Ling Chen. 2019. Semi-supervised Adversarial Active Learning on Attributed Graphs. *arXiv preprint arXiv:1908.08169* (2019).
[15] Li Liu, William K. Cheung, Xin Li, and Lejian Liao. [n. d.]. Aligning Users across Social Networks Using Network Embedding. In *IJCAI 2016*. 1774–1780.
[16] Sofus A Macskassy. 2009. Using graph-based metrics with empirical risk minimization to speed up active learning on networked data. In *Proceedings of the 15th ACM SIGKDD*. 597–606.
[17] Eric Malmi, Aristides Gionis, and Evimaria Terzi. 2017. Active network alignment: a matching-based approach. In *Proceedings of the 2017 ACM CIKM*. 1687–1696.
[18] Tong Man, Huawei Shen, Shenghua Liu, Xiaolong Jin, and Xueqi Cheng. 2016. Predict anchor links across social networks via an embedding approach.. In *IJCAI*.
[19] Kaare Brandt Petersen et al. [n. d.]. The matrix cookbook. ([n. d.]).
[20] Yuxiang Ren, Charu C Aggarwal, and Jiawei Zhang. 2019. Meta diagram based active social networks alignment. In *2019 IEEE 35th ICDE*. IEEE, 1690–1693.
[21] Burr Settles. 2009. *Active learning literature survey*. Technical Report. University of Wisconsin-Madison Department of Computer Sciences.
[22] Lixin Shi, Yuhang Zhao, and Jie Tang. 2012. Batch mode active learning for networked data. *ACM TIST* 3, 2 (2012), 1–25.
[23] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2007. Pairwise Global Alignment of Protein Interaction Networks by Matching Neighborhood Topology. In *11th Annual International Conference, RECOMB*. Springer.
[24] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2008. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences* 105, 35 (2008), 12763–12768.
[25] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2008. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences* 105, 35 (2008), 12763–12768.
[26] Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of machine learning research* 2, Nov (2001), 45–66.
[27] Véronique Van Vlasselaer, Cristián Bravo, Olivier Caelen, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. 2015. APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems* 75 (2015), 38–48.
[28] Reza Zafarani and Huan Liu. 2013. Connecting users across social media sites: a behavioral-modeling approach. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 41–49.
[29] Jiawei Zhang and S Yu Philip. 2015. Multiple anonymized social networks alignment. In *2015 IEEE International Conference on Data Mining*. IEEE, 599–608.
[30] Si Zhang and Hanghang Tong. 2016. Final: Fast attributed network alignment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1345–1354.
[31] Si Zhang and Hanghang Tong. 2019. Attributed Network Alignment: Problem Definitions and Fast Solutions. *IEEE Trans. Knowl. Data Eng.* 31, 9 (2019), 1680–1692. https://doi.org/10.1109/TKDE.2018.2866440
[32] Si Zhang, Hanghang Tong, Ross Maciejewski, and Tina Eliassi-Rad. 2019. Multilevel Network Alignment. In *The World Wide Web Conference (WWW '19)*. Association for Computing Machinery, New York, NY, USA, 2344–2354.
[33] Si Zhang, Hanghang Tong, Yinglong Xia, Liang Xiong, and Jiejun Xu. 2020. NetTrans: Neural Cross-Network Transformation. In *The 26th ACM SIGKDD*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 986–996.
[34] Si Zhang, Hanghang Tong, Jiejun Xu, Yifan Hu, and Ross Maciejewski. 2019. ORIGIN: Non-Rigid Network Alignment. In *IEEE Big Data*. IEEE.
[35] 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*. 1485–1494.
[36] Erheng Zhong, Wei Fan, Junwei Wang, Lei Xiao, and Yong Li. 2012. Comsoc: adaptive transfer of user behaviors over composite social network. In *Proceedings of the 18th ACM SIGKDD*. 696–704.
[37] Feng Zhou and Fernando De la Torre. 2012. Factorized graph matching. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 127–134.
[38] Qinghai Zhou, Liangyue Li, Nan Cao, Lei Ying, and Hanghang Tong. 2019. ADMIRING: Adversarial multi-network mining. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1522–1527.