ADMIRING: Adversarial Multi-Network Mining

Qinghai Zhou*, Liangyue Li†, Nan Cao‡, Lei Ying§, Hanghang Tong*
*University of Illinois at Urbana-Champaign, †Amazon, ‡Tongji University, §University of Michigan, Ann Arbor
*{qinghai2, htong}@illinois.edu, †liliangy@amazon.com, ‡nan.cao@gmail.com, §leiying@umich.edu

***Abstract*—Multi-sourced networks naturally appear in many application domains, ranging from bioinformatics, social networks, neuroscience to management. Although state-of-the-art offers rich models and algorithms to find various patterns when input networks are given, it has largely remained nascent on how vulnerable the mining results are due to the adversarial attacks. In this paper, we address the problem of attacking multi-network mining through the way of deliberately perturbing the networks to alter the mining results. The key idea of the proposed method (ADMIRING) is effective influence functions on the Sylvester equation defined over the input networks, which plays a central and unifying role in various multi-network mining tasks. The proposed algorithms bear two main advantages, including (1) *effectiveness,* being able to accurately quantify the rate of change of the mining results in response to attacks; and (2) *generality,* being applicable to a variety of multi-network mining tasks (e.g., graph kernel, network alignment, cross-network node similarity) with different attacking strategies (e.g., edge/node removal, attribute alteration).**

I. INTRODUCTION

Multi-sourced networks naturally appear in many high-impact application domains, ranging from bioinformatics, social networks, neuroscience to management. For example, for protein function prediction, a classic method is to assign similarity to protein pairs by applying graph kernel over multiple protein networks [1]. For team management, [2] proposes to replace the unavailable individual in the team by recommending the best candidate who maximizes the similarity of the team networks before and after the replacement (i.e., team-context aware similarity). For financial fraud detection, [3] resorts to interactive subgraph matching to identify complex fraud schema, e.g., syntheticIDs, money laundry, etc.

To date, many sophisticated multi-network mining models and algorithms have been proposed (See Section V for a review). Although these methods are quite effective in identifying various patterns when the input networks are given, less is known on how the mining results would be affected by the perturbation of the underlying networks, due to either random noise (i.e., sensitivity analysis) or malicious attacks (i.e., adversarial learning). For example, although graph kernel is effective in predicting the function (i.e., labels) of a protein, it is not clear how sensitive the prediction result is due to the measurement error for certain molecule-molecule interactions (i.e., edge error).

In this paper, we address the problem of attacking multi-network mining to alter its results, which we formulate as an optimization problem. We propose a family of algorithms (ADMIRING) to achieve effective attacks. Figure 1 presents one illustrative example of adversarial multi-network mining. The key idea behind our method is to quantitatively characterize how the mining result will change if we deliberately perturb the networks, e.g., editing the network topology by removing edges/nodes or modifying attributes. To be specific, given the central and unifying role of the Sylvester equation defined over the input networks in a variety of multi-network mining tasks (e.g., graph kernel, network alignment, etc.) [4], [5], we measure the influence of network elements (i.e., edge, node and attribute) as the rate of change of the mining results induced by the Sylvester equation.

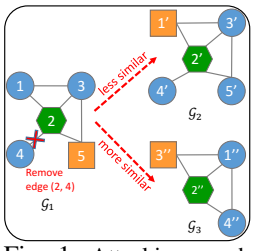

Fig. 1: Attacking graph kernel: given three networks $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$, $\mathcal{G}_1$ is much more similar to $\mathcal{G}_2$. By removing edge $(2, 4)$, the new network $\mathcal{G}_1'$ becomes more similar to $\mathcal{G}_3$ than it is to $\mathcal{G}_2$.

We summarize the main contributions of this paper as follows,

- **Problem Formulation.** We formally define the adversarial multi-network mining problem and formulate it as an optimization problem. The key idea is to measure the influence of network elements as the rate of change of the mining results induced by the underlying Sylvester equation.
- **Algorithms and Analysis.** We propose a family of algorithms (ADMIRING) to effectively solve the adversarial multi-network mining problem, which are applicable to a variety of multi-network mining tasks.
- **Empirical Evaluations.** We perform extensive experimental evaluations on real-world datasets to test the efficacy of our proposed algorithm. Our evaluations demonstrate that the algorithms can significantly alter the similarity between networks, the accuracy of network classification.

The rest of the paper is organized as follows. In Section II, we define the problem of adversarial multi-network mining. Section III introduces our proposed algorithms. We present experimental results in Section IV, and review related work in Section V. We conclude the paper in Section VI.

II. PROBLEM DEFINITIONS

In this section, we formally define adversarial multi-network mining problem, after we introduce notations and preliminaries on multi-network mining as well as influence function.

A. Notations

Table I summarizes the main symbols and notations used in this paper. We use bold uppercase letters for matrices (e.g., $\mathbf{A}$), bold lowercase letters for vectors (e.g., $\mathbf{q}$) and lowercase

*§ The work was partly done while the authors were at Arizona State University.

letters for scalars (e.g., c). For matrix indexing, we use $\mathbf{A}(i,j)$ to represent the entry at the i^{th} row and the j^{th} column of matrix $\mathbf{A}$, $\mathbf{A}(i,:)$ to denote the i^{th} row of $\mathbf{A}$ and $\mathbf{A}(:,j)$ to denote the j^{th} column of $\mathbf{A}$.

In this paper, we focus on a pair of node attributed networks, represented as $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{N}_1\}$ and $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{N}_2\}$, where $\mathbf{A}_t$ (t= 1, 2) represents the adjacency matrices of the input networks. $\mathbf{N}_t^j = \text{diag}(\mathbf{N}_t(:,j))(t = 1,2 \text{ and } j = 1,\ldots,d)$ represents the strength of all nodes having the j^{th} attribute in network $\mathcal{G}_t$. The uppercase bold letter $\mathbf{N}_\times$ is the combined node attribute matrix of the two networks $\mathbf{N}_\times = \sum_{j=1}^{d} \mathbf{N}_1^j \otimes \mathbf{N}_2^j$. For simplicity, we assume the input networks are (a) unweighted, (b) undirected and (c) of the same size. The generalization of the proposed method to weighted and/or directed networks of different sizes is straightforward.

Symbols	Definitions
$\mathcal{G} = \{\mathbf{A}, \mathbf{N}\}$	an attributed network
$\mathbf{A}$	adjacency matrix
$\mathbf{A}_\times$	Kronecker product of $\mathbf{A}_1$ and $\mathbf{A}_2$
$\mathbf{N}^l$	diagonal matrix of the l^{th} node attribute
$\mathbf{N}_\times$	combined node attribute matrix
$\mathbf{A}^{-1}$, $\mathbf{A}'$	inverse and transpose of matrix $\mathbf{A}$
$\mathbf{S}^{i,j}$	single entry matrix $\mathbf{S}^{i,j}(i,j) = 1$ and zeros elsewhere
$\mathbf{I}$	an identity matrix
c, $\alpha \in (0,1)$	a regularization parameter, damping factor
$\mathbf{p}_\times, \mathbf{q}_\times$	initial and stopping probability distribution
n, m	number of nodes and edges, respectively
d	dimension of node attribute vector
$\mathcal{I}(g)$	influence function of network element g
$\otimes$	Kronecker product
$a = \text{vec}(\mathbf{A})$	vectorize a matrix $\mathbf{A}$ in column order
$\mathbf{X} = \text{mat}(\mathbf{x}, n, n)$	reshape $\mathbf{x}$ to an $n \times n$ matrix in column order
$\mathbf{Y} = \text{diag}(\mathbf{y})$	diagonalize a vector $\mathbf{y}$

TABLE I: Symbols and Definition

B. Preliminaries

We briefly review (1) Sylvester equation for multi-network mining tasks, and (2) influence function for machine learning.

1) - Sylvester equation for multi-network mining. A unifying cornerstone behind many multi-network mining tasks can be attributed to the Sylvester equation defined over the input networks. In detail, given two node-attributed networks $\mathcal{G}_1$ and $\mathcal{G}_2$, we have the following generalized Sylvester equation

$$\mathbf{X} = \sum_{l=1}^{d} c\mathbf{M}_l\mathbf{X}\mathbf{T}_l' + \mathbf{B} \tag{1}$$

where c is a regularization parameter, $\mathbf{M}_l = \mathbf{N}_2^l\mathbf{A}_2, \mathbf{T}_l = \mathbf{N}_1^l\mathbf{A}_1$, and $\mathbf{B} \in \mathbb{R}^{n\times n}$ encodes the prior knowledge of the mining tasks. For instance, in network alignment [5], $\mathbf{B}$ is the preference matrix to encode anchor links; and in random walk graph kernel [6], $\mathbf{B}$ represents the initial probability distribution of the random walks on the direct product matrix. In Eq. (1), $\mathbf{X} \in \mathbb{R}^{n\times n}$ is the solution matrix. A numerical solution of the Sylvester equation usually costs at least $O(n^3)$ in time complexity [7], and some recent works [8], [9] are able to reduce the time complexity to be linear w.r.t. the input network size. By the Kronecker product properties, we have the following equivalent linear equation of Eq. (1),

$$\mathbf{x} = c\mathbf{N}_\times\mathbf{A}_\times\mathbf{x} + \mathbf{b} \tag{2}$$

where $\mathbf{x} = \text{vec}(\mathbf{X})$, $\mathbf{b} = \text{vec}(\mathbf{B})$ and $\mathbf{A}_\times = \mathbf{A}_1 \otimes \mathbf{A}_2$. The closed-form solution of $\mathbf{x}$ is given by $\mathbf{x} = (\mathbf{I} - c\mathbf{N}_\times\mathbf{A}_\times)^{-1}\mathbf{b}$.

Remarks. For clarity of the description of the proposed algorithms and analysis, we will mainly focus on a pair of node-attributed networks. Nonetheless, the proposed algorithms can be naturally generalized to handle other scenarios. For example, both Eq. (1) and Eq. (2) can be generalized to handle edge attributes as well [5], [9]. If the node and edge attribute information is absent, Eq. (1) degenerates to $\mathbf{X} = c\mathbf{A}_2\mathbf{X}\mathbf{A}_1' + \mathbf{B}$. If there are multiple (more than two) input networks, we can organize them into a combined network by matrix direct sum $\mathcal{G} = \{\mathbf{A}, \mathbf{N}\}$, where $\mathbf{A}$ and $\mathbf{N}$ are block diagonal matrices and each diagonal block represents one input network. Then Eq. (1) is defined w.r.t. the combined network $\mathcal{G}$ (i.e., $\mathcal{G} = \mathcal{G}_1 \bigoplus \mathcal{G}_2$).

It turns out the solution matrix $\mathbf{X}$ and its vectorization $\mathbf{x}$ encodes rich information of the input networks, and therefore have been used for a variety of mining tasks. For example, the random walk based graph kernel [6] is essentially a summation of all the entries of $\mathbf{X}$ linearly weighted by the stopping probability distribution $\mathbf{q}_\times'$ of random walks on the direct product matrix; the solution matrix $\mathbf{X}$ indicates the soft node-alignment between the input networks and the specific entries indicates the similarity or proximity between two nodes across networks (i.e., cross-network node proximity) [5]; the solution matrix $\mathbf{X}$ defined over a query network and a data network can be further fed into a goodness function [3] for subgraph matching. Conceptually, we can represent all the above multi-network mining results induced by the solution matrix $\mathbf{X}$ by a function f. For example, $f(\mathbf{X}) = \mathbf{q}_\times'\text{vec}(\mathbf{X})$ for random walk graph kernel [4]; $f(\mathbf{X}) = \mathbf{X}$ for (soft) network alignment [5]. Table II presents a summary for different choices of $f(\cdot)$ for multi-network minignt tasks.

2) - Influence function for machine learning. Influence function is a powerful analytical tool from robust statistics to evaluate the dependence of the estimator on the value of the data points [10]. The seminal work by Pang et al. [11] proposes to leverage influence function to assess the effect of each training example on the performance of the machine learning system, as a key step towards explainable machine learning. Its key idea is to trace the learning model's predictions back to the input training examples. In order to identify the training examples that are most responsible to model's behavior, they use influence function in accordance with the model that reflects how the learning model's parameters are affected if a training example is perturbed by an imperceptible amount.

C. Problem Definition

Generally speaking, adversarial learning aims to maximally alter the learning results by manipulating a small number of the input data points. In the case of multi-network mining, it translates to a small number of network elements (e.g., edges/nodes/attributes). Given the unifying role of the solution matrix $\mathbf{X}$ of Eq. (1), we formally define the adversarial multi-network mining problem as follows,

Problem 1. Adversarial Multi-network Mining

Given: (1) two input attributed networks $\mathcal{G}_1$ and $\mathcal{G}_2$, (2) the vectorization of the solution matrix $\mathbf{X}$ of Eq. (1), (3) a function $f(\mathbf{X})$ in Table II underlying the corresponding mining task, (4) an integer budget k, and (5) the specific network element type (i.e., `edge` vs. `node` vs. `attribute`);

Find: a set of k most influential network elements of the specified type so that $f(\mathbf{X})$ will change most if we attack (e.g., remove or alter) those elements.

Multi-network Mining Tasks	Function $f(\cdot)$
Random walk graph kernel [4]	$f(\mathbf{X}) = \mathbf{q}'_\times \text{vec}(\mathbf{X})$
Soft network alignment [5]	$f(\mathbf{X}) = \mathbf{X}$ or $f(\mathbf{X}) = \mathbf{vec}(\mathbf{X})$
Cross-network node similarity [5]	$f(\mathbf{X}) = \mathbf{X}(s,t)$
Subgraph matching [3]	$f(\mathbf{X}) = \text{argmin}_{\mathbf{M}}\ g(\mathbf{M}, \mathbf{X})$

TABLE II: Choices of functions $f(\cdot)$ w.r.t. the solution $\mathbf{X}$ of Sylvester Equation underlying various multi-network mining tasks. In cross-network node similarity, the similarity between node s in $\mathcal{G}_2$ and node t in $\mathcal{G}_1$ is $\mathbf{X}(s,t)$.

III. Algorithms and Analysis

In this section, we first formally formulate Problem 1 from the optimization perspective. Next, we derive the influence functions with respect to various network elements (e.g., edges, nodes and attributes) for multi-network mining tasks. Based on that, we propose effective and efficient algorithms which leverage such influence functions to attack multi-network mining results, together with some analysis.

A. Admiring *Formulation*

The intuition behind adversarial multi-network mining is to find a set of key network elements (e.g., edges, nodes, attributes) whose perturbation (e.g., removal, alteration) would cause the largest change of the function $f(\cdot)$ underlying a given mining task. For example, for random walk graph kernel, the goal of an adversarial attack is to significantly change the similarity (i.e., graph kernel) between two input networks by deliberately perturbing a small set of influential network elements. To be specific, let $\mathbf{X}$ be the original solution of Eq. (1) for the input networks $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathbf{X}_\mathcal{P}$ be the new solution matrix for networks $\mathcal{G}_{1\mathcal{P}}$ and $\mathcal{G}_2$ after we perturb the network elements in set $\mathcal{P}$. The corresponding mining results are $f(\mathbf{X})$ and $f(\mathbf{X}_\mathcal{P})$, respectively. We formally formulate Problem 1 as the following optimization problem,

$$\begin{aligned} \underset{\mathcal{P}}{\text{argmax}}\ \Delta f &= \left(f(\mathbf{X}) - f(\mathbf{X}_\mathcal{P})\right)^2 \\ \text{s.t. } |\mathcal{P}| &= k \end{aligned} \tag{3}$$

Note that for network alignment, the squared loss in the above formulation will be replaced by the L_2 norm if $f(\mathbf{X}) = \text{vec}(\mathbf{X})$ or the Frobenius norm if $f(\mathbf{X}) = \mathbf{X}$. In order to solve the above optimization problem, there are two crucial questions that need to be answered: (Q1) how to quantitatively evaluate the influence of a specific network element w.r.t. the function $f(\cdot)$ over the solution $\mathbf{X}$ of the Sylvester equation; and (Q2) how to leverage the influence function to identify a set of network elements to attack various multi-network mining tasks. In the next two subsections, we present our solutions to Q1 and Q2 according to the specific type of network elements (i.e., edges, nodes and attributes), respectively.

B. Network Element Influence

In order to achieve effective attacks to multi-network mining tasks, it is desirable that the change to the network structure would significantly impact the mining results (i.e., $f(\mathbf{X})$ in Table II). For simplicity and clarity, we only consider three types of attacks, including edge removal, node removal and node attribute alteration and we assume the attack always happens on the first network $\mathcal{G}_1$. In order to quantify how $f(\mathbf{X})$ changes (i.e., Δf in Eq. (3)) if we perturb a specific network element, we propose to use the influence function w.r.t. the corresponding multi-network mining tasks.

Definition 1. *Edge Influence in Multi-network Mining.* For a given multi-network mining task $f(\mathbf{X})$, the influence of a specific edge (e.g., $\mathbf{A}_1(i,j)$ in $\mathcal{G}_1$) w.r.t. the mining result is defined as the derivative of $f(\mathbf{X})$ w.r.t. this edge. Formally, the edge influence is defined as $\mathcal{I}(\mathbf{A}_1(i,j)) = \frac{\partial f(\mathbf{X})}{\partial \mathbf{A}_1(i,j)}$.

Definition 2. *Node Influence in Multi-network Mining.* The node influence is defined as the summation of influences of the incident edges, i.e., $\mathcal{I}(\mathbf{N}_1(i)) = \sum_{j|\mathbf{A}_1(i,j)=1} \mathcal{I}(\mathbf{A}_1(i,j))$

Definition 3. *Node Attribute Influence in Multi-network Mining.* For node-attributed networks, the influence of the l^{th} attribute of node i (i.e., $\mathbf{N}_1^l(i,i)$) in network $\mathcal{G}_1$ is defined as the derivate of $f(\mathbf{X})$ w.r.t. this specific node attribute, i.e., $\mathcal{I}(\mathbf{N}_1^l(i,i)) = \frac{\partial f(\mathbf{X})}{\partial \mathbf{N}_1^l(i,i)}$.

Next, we present details on how to compute the influence of network elements (e.g., edges, nodes and attributes) w.r.t. the mining results in the various tasks (e.g., random walk graph kernel, network alignment and cross-network node similarity in Table II). For clarity, we will mainly use random walk graph kernel as an example to illustrate the mathematical details to compute different network element influences.

1) Edge influence. We first give Lemma 1 to compute the edge influence for random walk graph kernel.

Lemma 1. (Edge Influence for Random Walk Graph Kernel.) Given random walk graph kernel between two input networks: $f(\mathbf{X}) = \mathbf{q}'_\times \left(\mathbf{I} - c\mathbf{N}_\times \mathbf{A}_\times\right)^{-1} \mathbf{N}_\times \mathbf{p}_\times$, the influence of a specific edge (e.g., $\mathbf{A}_1(i,j)$ in $\mathcal{G}_1$) w.r.t. random walk graph kernel can be calculated as follows,

$$\mathcal{I}(\mathbf{A}_1(i,j)) = c\mathbf{q}_\times{}'\mathbf{Q}\mathbf{N}_\times[(\mathbf{S}^{i,j} + \mathbf{S}^{j,i}) \otimes \mathbf{A}_2]\mathbf{Q}\mathbf{N}_\times\mathbf{p}_\times \tag{4}$$

where $\mathbf{Q} = \left(\mathbf{I} - c\ \mathbf{N}_\times\mathbf{A}_\times\right)^{-1}$, $\mathbf{S}^{i,j}$ is a single-entry matrix of the same size as $\mathbf{A}_1$, with 1 at the $(i,j)^{\text{th}}$ position and 0 elsewhere.

Proof. According to [4], the random walk graph kernel for node-attributed networks is,

$$f(\mathbf{X}) = \mathbf{q}'_\times \text{vec}(\mathbf{X}) = \mathbf{q}'_\times \left(\mathbf{I} - c\mathbf{N}_\times\mathbf{A}_\times\right)^{-1} \mathbf{N}_\times\mathbf{p}_\times \tag{5}$$

where $\text{vec}(\mathbf{X}) = \left(\mathbf{I} - c\mathbf{N}_\times\mathbf{A}_\times\right)^{-1} \mathbf{N}_\times\mathbf{p}_\times = \mathbf{x}$.

Following Definition 1, we take the partial derivative of $f(X)$ w.r.t. a specific edge (e.g., $\mathbf{A}_1(i,j)$ in network $\mathcal{G}_1$),

$$\mathcal{I}(\mathbf{A}_1(i,j)) = \frac{\partial f(\mathbf{X})}{\partial \mathbf{A}_1(i,j)} = \frac{\partial f(\mathbf{X})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{A}_1(i,j)} \tag{6}$$

According to the first row of Table II, we have that $\frac{\partial f(\mathbf{X})}{\partial \mathbf{x}} = \mathbf{q}'_\times$. For the second partial derivative (i.e., $\frac{\partial \mathbf{x}}{\partial \mathbf{A}_1(i,j)}$), by taking the derivative of Eq. (2), we have that

$$\frac{\partial \mathbf{x}}{\partial \mathbf{A}_1(i,j)} = c\left(\mathbf{I} - c\mathbf{N}_\times \mathbf{A}_\times\right)^{-1} \mathbf{N}_\times \frac{\partial \mathbf{A}_\times}{\partial \mathbf{A}_1(i,j)} \mathbf{x} \quad (7)$$

By the property of the matrix derivative [12, Page 8], we further have that

$$\frac{\partial \mathbf{A}_\times}{\partial \mathbf{A}_1(i,j)} = \frac{\partial \mathbf{A}_1}{\partial \mathbf{A}_1(i,j)} \otimes \mathbf{A}_2 = \left(\mathbf{S}^{i,j} + \mathbf{S}^{j,i}\right) \otimes \mathbf{A}_2 \quad (8)$$

Recall the closed-form solution $\mathbf{x} = \left(\mathbf{I} - c\mathbf{N}_\times \mathbf{A}_\times\right)^{-1} \mathbf{N}_\times \mathbf{p}_\times$. Putting everything together, we obtain the solution for calculating the influence of edge $\mathbf{A}_1(i,j)$ w.r.t. random walk graph kernel as follows,

$$\mathcal{I}\left(\mathbf{A}_1(i,j)\right) = c\mathbf{q}_\times{}'\mathbf{Q}\mathbf{N}_\times[(\mathbf{S}^{i,j} + \mathbf{S}^{j,i}) \otimes \mathbf{A}_2]\mathbf{Q}\mathbf{N}_\times \mathbf{p}_\times$$

which completes the proof. □

2) Node influence. Based on the edge influence (Eq. (4)), it is straight-forward to compute the node influence, which is summarized in the following proposition.

Proposition 1. (Node Influence in Random Walk Graph Kernel.) Given random walk graph kernel between two input networks: $f(\mathbf{X}) = \mathbf{q}'_\times \left(\mathbf{I} - c\mathbf{N}_\times \mathbf{A}_\times\right)^{-1} \mathbf{N}_\times \mathbf{p}_\times$, the influence of a specific node (e.g., $\mathbf{N}_1(i)$ in $\mathcal{G}_1$) w.r.t. random walk graph kernel can be calculated as,

$$\mathcal{I}\left(\mathbf{N}_1(i)\right) = c\mathbf{q}_\times{}'\mathbf{Q}\mathbf{N}_\times[\sum_{j|\mathbf{A}_1(i,j)=1} (\mathbf{S}^{i,j}+\mathbf{S}^{j,i})\otimes\mathbf{A}_2]\mathbf{Q}\mathbf{N}_\times\mathbf{p}_\times \quad (9)$$

Proof. It directly follows Lemma 1. Omitted for brevity. □

3) Node attribute influence. Finally, we give Lemma 2 to compute the node attribute influence.

Lemma 2. (Node Attribute Influence for Random Walk Graph Kernel.) Given random walk graph kernel between two input networks: $f(\mathbf{X}) = \mathbf{q}'_\times \left(\mathbf{I} - c\mathbf{N}_\times \mathbf{A}_\times\right)^{-1} \mathbf{N}_\times \mathbf{p}_\times$, the influence of a specific node attribute (e.g., the l^{th} dimension of the attribute vector of node $\mathbf{N}_1(i)$ in $\mathcal{G}_1$, i.e., $\mathbf{N}_1^l(i,i)$) w.r.t. random walk graph kernel can be calculated as follows,

$$\mathcal{I}\left(\mathbf{N}_1^l(i,i)\right) = \mathbf{q}'_\times \mathbf{Q}[\mathbf{S}^{i,i} \otimes \mathbf{N}_2^l]\left(\mathbf{I} + c\mathbf{A}_\times \mathbf{Q}\mathbf{N}_\times\right)\mathbf{p}_\times \quad (10)$$

where $\mathbf{Q} = \left(\mathbf{I} - c\ \mathbf{N}_\times \mathbf{A}_\times\right)^{-1}$, $\mathbf{S}^{i,i}$ is a single-entry matrix of the same size as $\mathbf{A}_1$, with 1 at the $(i,i)^{\text{th}}$ position and 0 elsewhere. $\mathbf{N}_2^l$ represents the strength of all the nodes having the l^{th} attribute from the network $\mathcal{G}_2$.

Proof. Omitted for brevity. □

C. Proposed Algorithms

1) A generic algorithm for adversarial multi-network mining. Based on Lemma 1, 2 and proposition 1, we propose Algorithm 1 to identify the most influential network elements (i.e., edges, nodes, attributes) for random walk graph kernel. Note that Algorithm 1 provides a family of attacking algorithms based on the specific network element. We use different suffix to differentiate different attacking scenarios, i.e., ADMIRING-E, ADMIRING-N, ADMIRING-A for attacking edges, nodes and node attributes respectively.

The key idea of the proposed ADMIRING algorithm is that we iteratively attack one network element with the highest influence value (Step-5, 9, 16), remove or alter it from the network (Step-6, 10, 17) and recompute the influence functions for the remaining network elements. In Algorithm 1, the two column vectors $\mathbf{p}_\times$ and $\mathbf{q}_\times$ are set as uniform unit vectors, i.e., $\mathbf{p}_\times = \mathbf{q}_\times = \mathbf{1} \times \frac{1}{n^2}$, and c is chosen as a small positive number to ensure convergence of the Sylvester Eq. (1) with fixed-point methods (e.g., $c = 1/(\max(\text{eigenvalue}(\mathbf{N}_1\mathbf{A}_1)) \times \max(\text{eigenvalue}(\mathbf{N}_2\mathbf{A}_2)) + 1)$).

Algorithm 1 ADMIRING: Adversarial Multi-network Mining

Input: (1) Two attributed networks $\mathcal{G}_1$ and $\mathcal{G}_2$, (2) an integer budget k, (3) a mining task denoted by $f(\cdot)$ in Table II, (4) network element type (i.e., `edge` vs `node` vs `node attribute`), (5) $\mathbf{q}_\times$, $\mathbf{p}_\times$, and (6) parameter c;
Output: A set of k network elements $\mathcal{P}$ to attack, and the residual network $\mathcal{G}_{1_\mathcal{P}}$.

1: Initialize $\mathcal{P} = \varnothing$;
2: **while** $|\mathcal{P}| < k$ **do**
3: **if** element type is `edge` **then**
4: Calculate influence for all edges using Eq. (4)
5: Add edge $(i,j) = \underset{(i,j)}{\text{argmax}}\, \mathcal{I}\left(\mathbf{A}_1(i,j)\right)$ to $\mathcal{P}$;
6: Remove edge (i,j) and (j,i) from $\mathcal{G}_1$;
7: **else if** element type is `node` **then**
8: Compute influence for all nodes in $\mathcal{G}_1$ by Eq. (9);
9: Add node $i = \underset{i}{\text{argmax}}\, \mathcal{I}\left(\mathbf{N}(i)\right)$ to $\mathcal{P}$;
10: Remove node i from $\mathcal{G}_1$;
11: **else if** element type is `node attribute` **then**
12: Set the value of damping factor $\alpha \in (0,1)$;
13: **for** node i in $\mathcal{G}_1$ **do**
14: Compute influence of each attribute by Eq. (10);
15: **end for**
16: Add node attribute $\underset{\mathbf{N}_1^l(i,i)}{\text{argmax}}\, \mathcal{I}\left(\mathbf{N}_1^l(i,i)\right)$ to $\mathcal{P}$;
17: Reduce the selected attribute by a ratio α in $\mathcal{G}_1$;
18: **else**
19: **return** Error
20: **end if**
21: **end while**
22: **return** $\mathcal{P}$ and $\mathcal{G}_{1_\mathcal{P}}$.

IV. EXPERIMENTAL EVALUATION

In this section, we conduct experiments to evaluate the proposed algorithms in terms of the following question:

- **Effectiveness.** How effective are the proposed algorithms in identifying key network elements to attack multi-network mining tasks?

A. Experimental Setup

Dataset	Class	Avg. #nodes	Avg. #edges	Attribute
MUTAGENICITY	2	30.32	19.79	0
PROTEINS	2	39.06	72.82	1
IMDB-BINARY	2	19.77	96.53	0

TABLE III: Statistics of datasets. The 'Class' column is the number of network labels. '0' in the 'Attribute' column means the corresponding networks do not have node attributes.

1) Datasets. We use three real-world datasets which are publicly available. Table III summarizes the statistics of the datasets. Detailed descriptions of these datasets are as follows.

- **MUTAGENICITY** is a dataset of 4,337 networks of molecular structures and it is divided into two classes *mutagen* (2,401 networks) and *nonmutagen* (1,936 networks) according to whether they have a property of mutagenicity [13].
- **PROTEINS** is a dataset of 1,113 protein structures [1]. Each protein is represented by a network. The task is to classify the protein structures into enzymes vs. non-enzymes.
- **IMDB-BINARY** is a movie collaboration dataset which collects cast and genre information of two types of movies, *romance* and *action* on IMDB. For each network, nodes represent actors/actresses and there is an edge between them if they appear in the same movie, and the task is to predict the genre of the movie the network represents [14].

2) Evaluation Metrics. We quantify the effectiveness of the proposed algorithms by measuring the following two aspects, including (1) the relative change of $f(\cdot)$ (i.e., $\frac{\Delta f}{f}$), and (2) the accuracy of a multi-network mining task (e.g., network classification) before and after performing adversarial attacks.

We perform the evaluations on random walk graph kernel, (1) on each selected dataset, we randomly selected 100 pairs of networks of the same class, then for every pair of networks, we apply ADMIRING or comparison methods to attack one of them, and re-compute the graph kernel to compare the relative change; (2) we also trained a SVM classifier with graph kernel for each of the three datasets; for every network (i.e., $\mathcal{G}_1$) in the testing set, we attacked it with ADMIRING or comparison methods to reduce its similarity with one random training network (i.e., $\mathcal{G}_2$) . The new classification result is from applying the trained SVM classifier on the attacked test networks. We report the average classification accuracy by repeating the above attacking strategy for 10 times.

3) Comparison Methods. We evaluate the proposed ADMIRING method with different attacking strategies for attacking edges, nodes and node attributes respectively. We also evaluate a batch-mode variant of ADMIRING. That is, in Algorithm 1, instead of selecting one network element at each iteration, we select the top-k elements with the highest influence scores in one iteration. We use a suffix 'v' to denote such a variant. For comparison, we use the following methods to select the top-k influential network elements, including (1) *Random*, which randomly select k network elements in the first network to attack; (2) *Bruteforce*, which re-computes f after attacking each element and selects the one with the highest Δf in each iteration; (3) *Bruteforce-v*, which uses *Bruteforce* in the batch mode; (4) *PageRank*, which measures the importance of nodes in a given network [15] and here we leverage the PageRank ranking results to select top-k influential network elements. In $\mathcal{G}_1$, we define the PageRank value of an edge $\mathbf{A}_1(i,j)$ as,

$$v(\mathbf{A}_1(i,j)) = (\mathbf{r}(i) + \mathbf{r}(j)) \times \max_{m \in \{i,j\}} \mathbf{r}(m)$$

where $\mathbf{r}(i)$ is the PageRank value of node i in $\mathcal{G}_1$. In ADMIRING-E, we choose the top k edges with the highest PageRank edge values; in ADMIRING-N, we select the k nodes in $\mathcal{G}_1$ with the largest PageRank values; and in ADMIRING-A, we first select the node with the the highest PageRank value and then select the attribute dimension with the largest value if the corresponding node attribute vector has multiple dimensions; (5) *Q-Matrix*, which selects the top-k network elements based on $\mathbf{Q} = (\mathbf{I} - c\mathbf{N}_\times\mathbf{A}_\times)^{-1}$. Recall that $\mathbf{Q}$ is an $n^2 \times n^2$ matrix and is closely correlated to the solution of the multi-network mining problems as shown in Table II. We can use a block-matrix representation (i.e., $\mathbf{Q} = [\mathbf{W}^{ij}]_{i,j=1,\ldots,n}$, where $\mathbf{W}^{ij}$ is at the $(i,j)^{\text{th}}$ position of $\mathbf{Q}$ with size $n \times n$). In edge attack, the aggregation of the entries in block matrix $\mathbf{W}^{ij}$ can be considered as the contribution of the edge $\mathbf{A}_1(i,j)$ to the mining result (i.e., $f(\mathbf{X})$). We can compute the influence of edge $\mathbf{A}_1(i,j)$ by the aggregation of the entries in $\mathbf{W}^{ij}$ and select the top-k edges. In node attack, we calculate the influence of node i in $\mathcal{G}_1$ by summing up all blocks $\mathbf{W}^{ij}|_{\mathbf{A}_1(i,j)=1}$ and select the top-k nodes. In node attribute attack, we compute the influence of $\mathbf{N}_1^l(i,i)$ as $\sum_{j=1}^n \mathbf{W}^{ij}(l,:)$ and select the top-k attributes.

B. Effectiveness Results

We compare the proposed ADMIRING with comparison methods in the task of attacking random walk graph kernel with different types of network elements, i.e., edges, nodes and attributes. The results of $\frac{\Delta f}{f}$ are summarized in Figures 2, 3, and 4, respectively. We can see that the proposed ADMIRING and its batch-mode variant ADMIRING-v (two red bars) achieve a very similar performance as their *bruteforce* counterparts (two yellow bars). As we will show in the next subsection, the *bruteforce* methods are much more expensive in terms of computation. In the meanwhile, the proposed methods (two red bars) consistently outperform all the remaining methods by a large margin in the three attacking scenarios across all datasets. For example, on *IMDB-BINARY* dataset, the proposed method ADMIRING-E is 4.31 times better than the best competitor method (i.e., ADMIRING-E vs. *Q-Matrix*) by attacking edges; on *MUTAGENICITY* dataset, by attacking nodes, the proposed ADMIRING-N is 269% better than *Q-Matrix*, and for node attribute attack, our proposed method ADMIRING-A is 2.36 times better than *Q-Matrix* on *PROTEINS* dataset. For the reported results in Figures 2, 3 and 4, the budget k is set to be 10, and the damping factor α is 0.2 for result in Figure 4.

Second, we evaluate and compare the impact of different attacking methods on the network classification results, summarized in Figure 5. We can see both the ADMIRING method and the *bruteforce* method can significantly impact the classification results. For example, on *PROTEINS* dataset, our proposed method can reduce the classification accuracy by 8.82%, 10.52%, and 6.98% by attacking edges, nodes and attributes, respectively. On the other hand, the attacking effect by other methods is quite marginal. For example, the best competitor method (*Q-Matrix*), can only achieve a reduction of 3.35% in classification accuracy by attacking nodes.

V. RELATED WORK

Multi-network Mining aims to collectively leverage the relationship among multiple networks for a better mining out-

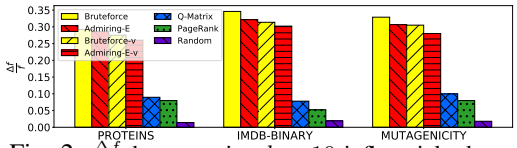

Fig. 2: $\frac{\Delta f}{f}$ by removing $k = 10$ influential edges.

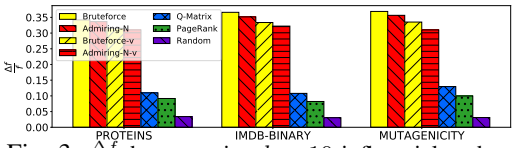

Fig. 3: $\frac{\Delta f}{f}$ by removing $k = 10$ influential nodes.

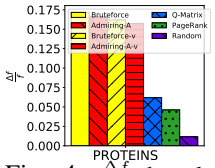

Fig. 4: $\frac{\Delta f}{f}$ by altering $k = 10$ influential node attributes ($\alpha = 0.2$).

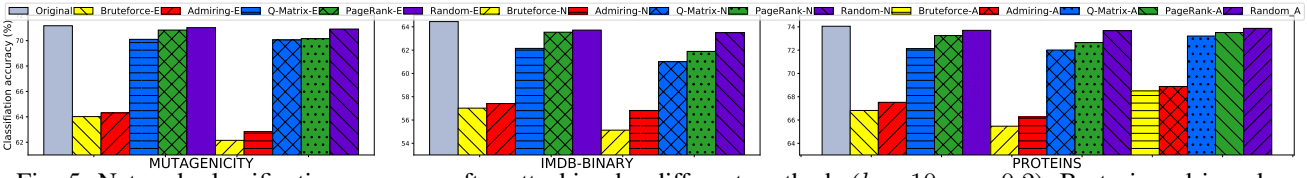

Fig. 5: Network classification accuracy after attacking by different methods ($k = 10, \alpha = 0.2$). Best viewed in color.

come or reveal patterns . Graph kernel is a family of methods that measure the similarity between two input networks based on walks [4], [6] or limited-sized subgraphs [16]. Network alignment aims to identify node correspondence across different networks. Some recent work suggests that the alignment can be enhanced by augmenting the topology consistency with the node and edge attributes [5], [8]. Multiple networks are manifested as a network of networks, where each node of the main network itself is another domain network [17], [18]. The main network can contextualize the mining tasks in each domain-specific network by providing the consistency constraints across networks for both ranking [17] and clustering [19]. With the increased complexity of the networked system, some efforts have been towards explainable network mining [20]–[22]. **Adversarial Learning** studies the learning process in the adversarial setting, where an adversary can attack the learning model [23]. In white-box evasion attack, the adversary has the full knowledge of the learning model (i.e., parameters and hyperparameters). The adversarial example is crafted by perturbing the most salient features. More recently, adversarial attacks on network mining algorithms start to receive attentions, where perturbations to the network attributes and structure can be constructed to mislead the graph convolution models [24], [25].

VI. CONCLUSION

In this paper, we study the problem of adversarial multi-network mining and formulate it as an optimization problem. The key idea is to effectively characterize the change of mining results w.r.t. the perturbations to the network. We propose a family of algorithms (ADMIRING) that are able to measure the influence of network elements to the mining results. The empirical evaluations on real-world datasets demonstrate the efficacy of the proposed algorithms. Future work includes generalizing the current method beyond Sylvester equation based solution in the black-box model setting as well as designing effective defensive strategies.

ACKNOWLEDGEMENT

This work is supported by NSF (IIS-1651203, IIS-1715385), DHS (2017-ST-061-QA0001) and Fundamental Research Funds for the Central Universities.

REFERENCES

[1] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl_1, pp. i47–i56, 2005.

[2] L. Li, H. Tong, N. Cao, K. Ehrlich, Y.-R. Lin, and N. Buchler, "Replacing the irreplaceable: Fast algorithms for team member recommendation," in *WWW*, 2015, pp. 636–646.

[3] B. Du, S. Zhang, N. Cao, and H. Tong, "First: Fast interactive attributed subgraph matching," in *KDD*. ACM, 2017, pp. 1447–1456.

[4] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *JMLR*, no. Apr, pp. 1201–1242, 2010.

[5] S. Zhang and H. Tong, "Final: Fast attributed network alignment," in *KDD*. ACM, 2016, pp. 1345–1354.

[6] K. M. Borgwardt, N. N. Schraudolph, and S. Vishwanathan, "Fast computation of graph kernels," in *NIPS*, 2007, pp. 1449–1456.

[7] J. D. Gardiner, A. J. Laub, J. J. Amato, and C. B. Moler, "Solution of the sylvester matrix equation axb t+ cxd t= e," *ACM Transactions on Mathematical Software (TOMS)*, vol. 18, no. 2, pp. 223–231, 1992.

[8] S. Zhang, H. Tong, J. Tang, J. Xu, and W. Fan, "ineat: Incomplete network alignment," in *ICDM*. IEEE, 2017, pp. 1189–1194.

[9] B. Du and H. Tong, "Fasten: Fast sylvester equation solver for graph mining," in *KDD*, ser. KDD '18. New York, NY, USA: ACM, 2018.

[10] P. J. Huber *et al.*, "The 1972 wald lecture robust statistics: A review," *The Annals of Mathematical Statistics*, pp. 1041–1067, 1972.

[11] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," *arXiv preprint arXiv:1703.04730*, 2017.

[12] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," nov 2012, version 20121115.

[13] J. Kazius, R. McGuire, and R. Bursi, "Derivation and validation of toxicophores for mutagenicity prediction," *Journal of Medicinal Chemistry*, vol. 48, no. 1, pp. 312–320, 2005.

[14] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *KDD*, 2015.

[15] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[16] T. Horváth, T. Gärtner, and S. Wrobel, "Cyclic pattern kernels for predictive graph mining," in *KDD*, 2004, pp. 158–167.

[17] J. Ni, H. Tong, W. Fan, and X. Zhang, "Inside the atoms: ranking on a network of networks," in *KDD*, 2014, pp. 1356–1365.

[18] C. Chen, J. He, N. Bliss, and H. Tong, "On the connectivity of multi-layered networks: Models, measures and optimal control," in *ICDM*, 2015.

[19] J. Ni, H. Tong, W. Fan, and X. Zhang, "Flexible and robust multi-network clustering," in *KDD*, 2015, pp. 835–844.

[20] L. Li, H. Tong, and H. Liu, "Towards explainable networked prediction," in *CIKM*, ser. CIKM '18, 2018.

[21] Q. Zhou, L. Li, N. Cao, N. Buchler, and H. Tong, "Extra: Explaining team recommendation in networks," in *RecSys*. ACM, 2018.

[22] J. Kang, M. Wang, N. Cao, Y. Xia, W. Fan, and H. Tong, "Aurora: Auditing pagerank on large graphs," in *Big Data*. IEEE, 2018, pp. 713–722.

[23] B. Li, Y. Wang, A. Singh, and Y. Vorobeychik, "Data poisoning attacks on factorization-based collaborative filtering," in *NIPS*, 2016.

[24] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *KDD*. ACM, 2018, pp. 2847–2856.
[25] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *ICML*. Stockholmsmssan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 1115–1124.