ABSTRACT
Recent studies have shown that graph learning models are highly vulnerable to adversarial attacks, and network alignment methods are no exception. How to enhance the robustness of network alignment against adversarial attacks remains an open research problem. In this paper, we propose a robust network alignment solution, RNA, for offering preemptive protection of existing network alignment algorithms, enhanced with the guidance of effective adversarial attacks. First, we analyze how popular iterative gradient-based adversarial attack techniques suffer from gradient vanishing issues and show a fake sense of attack effectiveness. Based on dynamical isometry theory, an attack signal scaling (ASS) method with established upper bound of feasible signal scaling is introduced to alleviate the gradient vanishing issues for effective adversarial attacks while maintaining the decision boundary of network alignment. Second, we develop an adversarial perturbation elimination (APE) model to neutralize adversarial nodes in vulnerable space to adversarial-free nodes in safe area, by integrating Dirac delta approximation (DDA) techniques and the LSTM models. Our proposed APE method is able to provide proactive protection to existing network alignment algorithms against adversarial attacks. The theoretical analysis demonstrates the existence of an optimal distribution for the APE model to reach a lower bound. Last but not least, extensive evaluation on real datasets presents that RNA is able to offer the preemptive protection to trained network alignment methods against three popular adversarial attack models.

KEYWORDS
Adversarial Perturbation Elimination; Network Alignment

ACM Reference Format:

1 INTRODUCTION
Network alignment (i.e., graph matching) is one of the most important research topics in the graph domain, which aims to match the same entities (i.e., nodes) across multiple networks (i.e., graphs) [13, 30, 45, 50, 72, 79]. It has been widely applied to many real-world applications ranging from protein network alignment in bioinformatics [35, 41], user account linking in multiple social networks [22, 38, 75, 76], and object matching in computer vision [24, 56], to knowledge translation in multilingual knowledge bases [63, 104].

Despite the remarkable performance of existing graph learning models on clean networks, recent studies have shown that many models are fairly sensitive to adversarial attacks, i.e., carefully designed small perturbations in graph structure and attributes. Many encouraging adversarial defense progresses have been made towards improving model robustness against adversarial attacks, including node classification [19, 20, 23, 34, 55, 56, 62, 71, 74, 78, 103], graph classification [32], community detection [31], network embedding [15], link prediction [77], malware detection [29], spammer detection [17], fraud detection [7, 70], and influence maximization [42]. However, the majority of existing techniques focus on the adversarial attacks and defenses on single graph learning tasks. Multiple graph learning is much more difficult to study since it needs to analyze both intra-graph and inter-graph interactions of multiple graphs. A recent study has demonstrated that graph matching (i.e., network alignment) methods are highly vulnerable to adversarial attacks [73]. It proposes to estimate and maximize the densities of nodes to be attacked, for pushing them to dense regions in two graphs to generate imperceptible and effective attacks. There is still a general lack of robust methods investigating how to make network alignment robust to adversarial attacks, which demands for new techniques to address the following critical challenges.

Most of the above adversarial defense approaches fall into two categories: (1) Adversarial training techniques generate adversarial perturbations on clean graph data and retrain the learning models on perturbed graph data, i.e., modify the architecture of the target models to adapt to change [15, 21, 33, 62]. With the guidance of generated adversarial perturbations, the adversarial training methods exhibit good robustness against adversarial attacks. However, the adversarial training on graph data is non-trivial since it needs to train the model on both clean and perturbed graphs. Running the adversarial training for the multiple graph learning tasks (e.g., network alignment) makes the defense methods more inefficient and thus limit their applicability; and (2) Attack detection/elimination approaches aim to detect and remove perturbations or reduce the negative effect of attacks without the model retraining [58, 103]. However, a recent literature reports that the lack of supervised information about effective perturbations in a poisoned graph obstructs models from detecting adversarial edges and thus leads to sub-optimal solutions [55]. Therefore, the authors proposed to perturb the clean graphs that serve as supervised knowledge to train the ability to detect adversarial edges such that the robustness of
GNNS is elevated. Can we leverage the strengths of both adversarial training and attack elimination to learn a preemptive protection model for robust network alignment, by using the effective adversarial attacks as the supervision while avoiding the retraining of network alignment algorithms?

Recently, iterative gradient-based adversarial attack techniques, such as Fast Gradient Sign Method (FGSM) [25], Projected Gradient Descent (PGD) technique [43], and their variants, have shown the strength of producing effective adversarial examples in image data along the direction of gradient ascent. These methods compute the gradient of the loss function of target model to identify the weakest input features to attack. A large number of research efforts in adversarial attacks on graph data utilize the iterative gradient-based methods to produce effective adversarial perturbations that fool a graph learning model [14, 54, 62, 105]. However, a recent work reports that the gradient-based adversarial attack methods tend to fail to produce effective adversarial perturbations in scenarios where the gradients are uninformative, i.e., vanishing gradients due to poor backward signal propagation in neural networks [3]. How to improve the attack signal propagation of neural networks for effective adversarial attacks without affecting the decision boundary of network alignment?

With these challenges in mind, this paper proposes a robust network alignment solution that produces effective adversarial attacks on network alignment and utilizes them as the supervision to eliminate the adversarial perturbations before feeding it into given network alignment methods for offering the preemptive protection.

In order to ensure informative attack signal with both well-conditioned Jacobian and meaningful signal propagation from the loss of network alignment, we analyze how poor signal propagation can cause vanishing gradients in adversarial attacks on network alignment, and then propose an attack signal scaling (ASS) method based on the dynamical isometry theory to scale attack signal in back-propagation. The proposed method can improve the effectiveness of gradient-based adversarial attack while not affecting the prediction (i.e., decision boundary) of the trained network alignment algorithms. We also conduct the theoretical analysis to establish the upper bound of feasible signal scaling.

By integrating Dirac delta approximation (DDA) techniques and the long short-term memory (LSTM) models, an adversarial perturbation elimination (APE) model is developed to neutralize adversarial nodes in vulnerable space to adversarial-free nodes in safe area, such that the original clean and adversarial-free networks are close to each other. Our APE method can be integrated with existing trained models to offer robust network alignment solutions. The theoretical analysis demonstrates the correlation between the defense loss on adversarial-free nodes and the original network alignment loss on clean nodes. We also exhibit the existence of an optimal distribution for the APE model to reach a lower bound.

Empirical evaluation over real network datasets demonstrates that the considerable robustness improvement of RNA for several representative network alignment algorithms against three popular attack models in graph adversarial training.

To our best knowledge, this work is the first to study adversarial perturbation elimination for robust network alignment.

## 2 PROBLEM STATEMENT

In this paper, we aim to learn a preemptive protection model for existing network alignment algorithms, enhanced with the guidance of effective adversarial attacks.

Given one source network $G^1 = (V^1, E^1)$ and one target network $G^2 = (V^2, E^2)$ to be aligned, each network is denoted as $G^t = (V^t, E^t)\ (t = 1 \text{ or } 2)$, where $V^t = \{v^t_1, \cdots, v^t_N\}$ is the set of $N^t$ nodes and $E^t = \{(v^t_i, v^t_j) : 1 \leq i, j \leq N^t\}$ is the set of edges. A node $v^t_i \in V^t$ (1 $\leq i \leq N^t$) represents an entity in $G^t$. An edge $(v^t_i, v^t_j) \in E^t$ is associated with two nodes $v^t_i \in V^t$ and $v^t_j \in V^t$ and denotes the relationship between two corresponding entities.

Each $G^t$ has an $N^t \times N^t$ binary adjacency matrix $A^t$, where each entry $A^t_{ik} = 1$ if there exists an edge $(v^t_i, v^t_k) \in E^t$; otherwise $A^t_{ik} = 0$. $A^t_i$ specifies the $i$th row vector of $A^t$. In this paper, if there are no specific descriptions, we use $v^t_j$ to denote a node $v^t_i$ itself and its representation $v^t_i$, i.e., $v^t_i = A^t_i$, and we utilize $v^t_{ik}$ to specify the $k$th dimension of $v^t_i$, i.e., $v^t_{ik} = A^t_{ik}$.

The dataset is divided into two disjoint sets $D$ and $D'$. The former denotes a set of known aligned node pairs $D = \{(v^1_i, v^2_j) : v^1_i \leftrightarrow v^2_j \in V^1, v^2_j \in V^2\}$, where $v^1_i \leftrightarrow v^2_j$ indicates that two nodes $v^1_i$ and $v^2_j$ belong to the same entity. The latter, denoted by $D' = \{(v^1_i, v^2_j) : v^1_i \leftrightarrow v^2_j, v^1_i \in V^1, v^2_j \in V^2\}$, is used to evaluate the network alignment performance, where the nodes (but not their alignments) are also observed during training. The goal of supervised network alignment is to use $D$ as the training data to identify the one-to-one matching relationships between nodes $v^1_i$ and $v^2_j$ belonging to the same entities in the test data $D'$.

Many supervised learning methods learn effective network alignment algorithms by maximizing the similarities (or minimizing the distances) between projected source anchor nodes $M(v^1_i) \in D$ and target ones $v^2_j \in D' [38, 39, 64, 76]$. The node pairs $(v^1_i, v^2_j) \in D'$ with the largest similarities are selected as the alignment results. The following loss function is minimized to learn an injective one-to-one matching function $M : v^1_i \in V^1 \mapsto v^2_j \in V^2$. In deep network alignment approaches [13, 24, 39, 76], $M$ is often implemented as a neural network.

$$
\begin{align*}
\min_M \mathcal{L} &= \mathbb{E}_{(v^1_i, v^2_j) \in D} \mathcal{L}(v^1_i, v^2_j) \\
&= \mathbb{E}_{(v^1_i, v^2_j) \in D} \mathcal{L}(v^1_i, v^2_j) \\
&= \sum_{k=1}^{K} \mathbb{E}_{v^1_i \sim p(v^1_i)} \log \sigma(M(v^1_i)^T \cdot v^2_j)
\end{align*}
$$

where $M(v^1_i)^T$ is the transpose of $M(v^1_i)$, $p(v^2_j)$ denotes the distribution for sampling $K$ negative nodes $v^2_k \neq v^2_j$ through the negative sampling method [44], $\sigma(*)$ is the sigmoid function. The inner product $\cdot$ represents the similarity degree between two node vectors. The above loss is equivalent to a cross entropy loss with $(v^1_i, v^2_j) \in D$ as positive samples and $(v^1_i, v^2_j) \notin D$ as negative ones.

Given a trained network alignment method $v^1_i = M(v^1_i)$, an adversarial attacker aims to maximally degrade the alignment performance of $M$ on the test data $D'$ by injecting edge perturbations (including edge insertion and deletion) into $G^t = (V^t, E^t)\ (t = 1$
or 2), leading to two adversarial networks \( \hat{G}_1 = (\hat{V}_1, \hat{E}_1) \). In order to generate effective adversarial perturbations for better training adversarial perturbation elimination, we assume that the attacker can access the prediction result and gradient information of \( M \).

In contrast, with the generated adversarial perturbations as the supervision, an adversarial defender is trained to be able to eliminate the perturbations before feeding it into \( M \) for providing the preemptive protection. A desired perturbation elimination result should ensure that the model achieve the high utility of network alignment on the newly perturbed networks \( \hat{G}_1 \) and \( \hat{G}_2 \).

3 OVERVIEW

This paper proposes a robust network alignment solution that contains two analytics components: (1) Adversarial attacks with attack signal scaling and (2) Adversarial perturbation elimination via Dirac delta approximation, as shown in Figure 1. Here, given two networks \( G_1 \) and \( G_2 \), two nodes within each network are close/distant if they have similar/dissimilar structural features. Two red dots denote a pair of aligned nodes \( v_1 \) and \( v_2 \) in \( G_1 \) and \( G_2 \). The red dots are target nodes to be attacked.

(1) Adversarial attacks with attack signal scaling model attempts to generate the effective adversarial nodes in \( G_1 \) and \( G_2 \) that can easily fool the network alignment algorithm \( M \) trained on clean \( G_1 \) and \( G_2 \), and thus output wrong alignment results: (a) An attack signal scaling (ASS) method based on the dynamical isometry theory is proposed to compute signal scales \( \alpha_1 \) and \( \alpha_2 \) by using Eq.(6) and Theorem 5.1, in order to ensure informative attack signal with both well-conditioned Jacobian and meaningful signal propagation from the alignment loss in Eq.(1); (b) By integrating scaled signal \( \hat{M}(v_1^j) = \alpha_1\alpha_2M(v_1^j) \) and gradients \( \frac{\partial \hat{M}(v_1^j)}{\partial v_1^j} \) and \( \frac{\partial \hat{M}(v_1^j)}{\partial \hat{v}_1^j} \), the Projected Gradient Descent (PGD) is utilized to add and remove noisy edges to \( v_1^j \) and \( v_2^j \) in terms of Eq.(2) and Algorithm 1. The attacker moves \( v_1^j \) and \( v_2^j \) and derives adversarial nodes \( \hat{v}_1^j \) and \( \hat{v}_2^j \), such that the similarity \( \log \sigma(M(\hat{v}_1^j)^T, \hat{v}_2^j) - \log \sigma(M(v_1^j)^T, v_2^j) \) > \( \log \sigma(M(v_1^j)^T, v_2^j) \), where \( v_1^j \neq v_2^j \) is a negative node, and thus a wrong alignment \( \hat{v}_1^j = \hat{M}(\hat{v}_1^j) \) is produced.

(2) Adversarial perturbation elimination via Dirac delta approximation tries to rule out the negative effects of adversarial nodes and improve the robustness on the perturbed networks: (a) An adversarial perturbation elimination (APE) model with the LSTM models is proposed to neutralize adversarial nodes \( v_1^j \) (or \( v_2^j \)) in vulnerable space to adversarial-free nodes \( \hat{v}_1^j \) (or \( \hat{v}_2^j \)) in safe area in Eq.(8); (b) A Dirac delta approximation (DDA) technique is designed to make \( \hat{v}_1 \) (or \( \hat{v}_2 \)) close to \( v_1^j \) (or \( v_2^j \)) as much as possible, as well as cause the defense loss \( L_D \) on adversarial-free nodes in Eq.(7) to be identical to the original network alignment loss \( L \) on clean nodes in Eq.(1); (c) The adversarial-free \( \hat{v}_1^j \) and \( \hat{v}_2^j \) are fed into the trained \( M \) to output the network alignment results.

4 ADVERSARIAL ATTACKS WITH ATTACK SIGNAL SCALING

In this section, we will analyze how poor signal propagation can cause vanishing gradients in iterative gradient-based adversarial attacks, and then propose an attack signal scaling (ASS) method based on the dynamical isometry theory to scale attack signal in back-propagation, to ensure informative attack signal with both well-conditioned Jacobian and meaningful signal propagation from the alignment loss.

4.1 PGD-based Adversarial Attacks

Based on the alignment loss in Eq.(1), we propose to utilize the Projected Gradient Descent (PGD) method to produce adversarial nodes towards network alignment.

\[
\begin{align*}
(v_1^j)^{(s+1)} &= \Pi_{\Delta_1} \text{sgn}(\text{ReLU}(\nabla(v_1^j)^T \cdot L((v_1^j)^T, (v_2^j)^T))) \\
(v_2^j)^{(s+1)} &= \Pi_{\Delta_2} \text{sgn}(\text{ReLU}(\nabla(v_2^j)^T \cdot L((v_1^j)^T, (v_2^j)^T))),
\end{align*}
\]

where \((v_1^j)^T\) and \((v_2^j)^T\) denotes the adversarial nodes of \( v_1^j \) and \( v_2^j \) derived at step \( s \). \( \epsilon \) specifies the budget of allowed perturbed edges for each attacked node. \( \Delta_1 = \{\|\delta_1^T\| \leq \epsilon \} \) and \( \Delta_2 = \{\|\delta_2^T\| \leq \epsilon \} \) represents the constraint set of the projection operator \( \Pi \), i.e., it encodes whether an edge of \( v_1^j \) is modified or not. \( \Delta_1^T \) has the similar definition for \( v_2^j \). The composition of the ReLU and sign operators guarantees \((v_1^j)^T \in \{0, 1\}^N\) and \((v_2^j)^T \in \{0, 1\}^N\), as it adds (or removes) an edge or keeps it unchanged when an derivative in the gradient is positive (or negative). The outputs \((v_1^j)^S\) and \((v_2^j)^S\) at final step \( S \) are used as the adversarial nodes \( \hat{v}_1 \) and \( \hat{v}_2 \), i.e., \( \hat{v}_1 = (v_1^j)^S \) and \( \hat{v}_2 = (v_2^j)^S \).
Recall the alignment loss in Eq.(1), the gradient $\nabla_{y_j} L(y_j, y_j')$ in the PDG is mainly determined with $\frac{\partial}{\partial v_k} \log \sigma(M(v_j')^T \cdot v_k)$, where $v_k$ can be either positive sample $v_1$ and negative ones $v_k$ in Eq.(1). $
abla_{y_j} L(y_j, y_j')$ is a function of $M(v_j')$ and $M(v_j')$ is a function of $v_j$. We employ the chain rule for composite functions to compute the gradient of $\log \sigma(M(v_j')^T \cdot v_k)$.

$$\frac{\partial}{\partial v_i} \left( \log \sigma(M(v_j')^T \cdot v_k) \right) = \frac{\partial}{\partial v_i} \left( \log \sigma(M(v_j')^T \cdot v_k) \right) \frac{\partial M(v_j')^T}{\partial v_i}$$

(3)

where $\phi(M(v_j')^T, v_k)$ is a signal from the alignment loss and $J_i = \frac{\partial M(v_j')^T}{\partial v_i} \in \mathbb{R}^{N \times N}$ is the input-output Jacobian matrix of the neural network. Both the signal $\phi(M(v_j')^T, v_k)$ and the Jacobian matrix $J_i$ influence the gradient $\frac{\partial}{\partial v_i} \left( \log \sigma(M(v_j')^T \cdot v_k) \right)$ together. The attack would fail if either the signal has saturating gradient or the Jacobian is poorly conditioned. Either of them will lead to vanishing gradients.

4.2 Signal Scaling via Dynamical Isometry

A recent study describes that a neural network is dynamical isometry if all singular values $\lambda_{ir}$ of the Jacobian $J_i$ are close to 1, i.e., $1 - \lambda_{ir} \leq \xi$ for all $r \in \{1, \ldots, \min(N^2, N^2)\}$ and a small positive number $\xi = 0$ [48]. In this case, the loss signal $\phi(M(v_j')^T, v_k)$ backpropagates isometrically through the neural network, and thus maintains the norm and all angles between two vectors. They utilize the dynamical isometry to speed up the training of neural networks by improving the signal propagation. Motivated by this, we explore to employ the dynamical isometry to improve the iterative gradient-based adversarial attack.

In order to keep the decision boundary of network alignment invariant, we propose to discover a well-chosen scalar $\alpha \geq 0$ to improve signal propagation as well as maintain the relative order of the logits at output layer of the neural network.

$$\tilde{M}(v_1) = \alpha M(v_1)$$

(4)

where $\alpha$ is the attack signal scale.

By enhancing with $\alpha$, the gradients can be reformulated below.

$$\frac{\partial}{\partial v_i} \left( \log \sigma(\tilde{M}(v_j')^T \cdot v_k) \right) = \frac{\partial}{\partial v_i} \left( \log \sigma(M(v_j')^T \cdot v_k) \right) \frac{\partial \tilde{M}(v_j')^T}{\partial v_i} \frac{\partial M(v_j')^T}{\partial v_i}$$

(5)

where $\alpha$ linearly influences the Jacobian matrix $J$ while nonlinearly affecting the the loss signal $\phi(M(v_j')^T, v_k)$.

---

**Algorithm 1 PGD Attacks with Attack Signal Scaling**

**Input:** Source network $G_1 = (V_1, E_1)$, target network $G_2 = (V_2, E_2)$, set of known aligned node pairs $D = \{(v_1^j, v_2^j)\}$, trained network alignment model $M$, noise budget $e$, and signal threshold $\eta$.

**Output:** Adversarial node pairs $\{(v_1^s, v_2^s)\} \in D$.

1. for each aligned node pair $(v_1^j, v_2^j)$ in $D$
2. Initialize $v_1^1 = v_1^i$ and $v_2^j = v_2^j$.
3. Compute signal scale $a_1 = \frac{|D|}{\sum_{i=1}^{\lambda} 1_{\text{ratio}}} \cdot \eta$ in Eq.(6);
4. for $s = 1, \ldots, S$
5. Initialize signal scale $a_s = 1.0$;
6. for each $v_2^i$ of positive sample $v_2^i$ and $K$ negative ones $v_k^i$
7. if $1 - \sigma(\alpha M(v_1^j)^T \cdot v_k^i) \leq \eta/\|v_2^i\|_2$
8. Update $a_s = 1 - \frac{1}{2} \log \frac{1}{e}$ based on Theorem 5.1;
9. Scale attack signal $M(v_1^j) = a_s M(v_1^j)$;
10. Calculate $\frac{\partial}{\partial v_i} \phi(\tilde{M}(v_1^j)^T \cdot \tilde{v}_k)$ and $\frac{\partial}{\partial v_i} \phi(\tilde{M}(v_1^j)^T \cdot \tilde{v}_k)$;
11. Use the PGD to update $\tilde{v}_1$ in terms of Eq.(2);
12. Return $\{(v_1^s, v_2^s)\} \in D$.

As $\phi(\tilde{M}(v_1^j)^T, v_k^i) = \frac{\sigma(\tilde{M}(v_1^j)^T \cdot \tilde{v}_k)}{\|\tilde{M}(v_1^j)^T \cdot \tilde{v}_k\|_2}$, it is clear that the loss signal is equal to zero when $\alpha = \infty$ in Eq.(5). This leads to the vanishing gradient issue in adversarial attacks. On the other hand, when $\alpha = 0$, the norm of the loss signal is maximal. However, the singular values of $\alpha J_i$ are all zeros. Thus, $\frac{\partial}{\partial v_i} \phi(\tilde{M}(v_1^j)^T, v_k^i)$ is equal to zero, which results in the vanishing gradient issue too.

Based on the above analysis, in order to alleviate the vanishing gradients, a desired $\alpha$ should (1) guarantee $a_s$ is well conditioned, i.e., all singular values of $a_s J_i$ are close to 1, such that the loss signal $\phi(M(v_1^j)^T, v_k^i)$ can be well backpropagated from the output layer to the input layer and (2) ensure the loss signal meaningful, i.e., $\|\phi(M(v_1^j)^T, v_k^i)\|_2 > \eta$ for a certain positive signal threshold $\eta$.

First, we choose $\alpha$ as the inverse of the mean of singular values of $J_i$, so as to scale the mean of singular values of $\alpha J_i$ to closer to 1.

$$\alpha = \frac{|D| N}{\sum_{i=1}^{|D|} \sum_{r=1}^{N^2} \lambda_{ir}}$$

(6)

where $|D|$ is the size of the set $D$ of aligned node pairs and $N = \min(N^2, N^2)$, $\lambda_{ir}$ denotes the $r^{th}$ singular value of $J_i$.

Second, in order to guarantee $\|\phi(M(v_1^j)^T, v_k^i)\|_2 > \eta$, we need to select $\alpha$ to ensure $\left(1 - \sigma(\tilde{M}(v_1^j)^T \cdot \tilde{v}_k)\right) > \eta/\|v_2^i\|_2^2$. The following theorem demonstrates the upper bound of a qualified $\alpha$.

**Theorem 4.1.** Let nodes $v_j$ and $v_k$ be the most similar and least similar to $M(v_1^j)^T$ ($1 \leq j, k \leq N^2$), i.e., $v_j = \text{argmax}_{j} (M(v_1^j)^T \cdot v_j^2)$ and $v_k = \text{argmin}_{k} (M(v_1^j)^T \cdot v_k^2)$. Also,
suppose that $d$ is the minimal norm of node representation vectors in $G^3$, i.e., $d = \min_\eta \| v_0^\eta \|_2$ for $\forall v_0^\eta \in V^2$. For a given $0 < \eta < d/2$, if $\alpha > \frac{1}{c} \log \frac{d - \eta}{\eta}$, then $1 - \sigma(\alpha M(v_1^j)T \cdot v_2^j) > \| v_1^j \|_2$ for $\forall v_1^j \in V^2$.

Proof. Please refer to Appendix A.1 for detailed proof.

The above two types of $\alpha$ are integrated together to make the Jacobian well-conditioned as well as the loss signal meaningful.

By aggregating PGD-based adversarial attacks and attack signal scaling together, Algorithm 1 presents the pseudo code of our adversarial attack model. Line 2 initializes adversarial nodes $v_1^j$ and $v_2^j$ with real aligned nodes $v_1^j$ in $G^1$ and $v_2^j$ in $G^2$. Line 3 computes the first signal scale $a_1$ to make the Jacobian $a_1$ well-conditioned. Lines 5-8 calculate the second signal scale $a_2$ to make the loss signal meaningful. Lines 9-10 scale and improve the attack signal propagation of neural networks by integrating two signal scales. Line 11 utilizes the PGD method to add and remove edges. The loop repeats the above iterative procedure until achieving the maximum iterations of the PGD.

5 ADVERSARIAL PERTURBATION ELIMINATION

In this section, we develop an adversarial perturbation elimination (APE) model by integrating Dirac delta approximation (DDA) techniques and the LSTM models to offer preemptive protection to trained network alignment models.

5.1 Perturbation Elimination

Based on a trained network alignment model $M$ and any network alignment loss, e.g., the loss $L$ defined in Eq.(1), the defender aims to learn an APE model $P(\hat{v}_l^j|v_l^j)$ (or $Q(\hat{v}_l^j|v_l^j)$) to neutralize adversarial nodes $v_1^j$ (or $v_2^j$) in vulnerable space $A_1^j$ (or $A_2^j$) to adversarial-free nodes $v_1^j$ (or $v_2^j$) in safe area, such that $v_1^j$ (or $v_2^j$) are close to original clean nodes $V_1^j$ (or $V_2^j$). The defense loss function $L_D$ is defined to minimize the following marginalized expectation:

$$L_D = \mathbb{E}_{(v_1^j, v_2^j) \in D} \int_{A_1^j} \int_{A_2^j} \mathbb{E}_{v_l^j \sim P(\hat{v}_l^j|v_l^j)} \mathbb{E}_{v_2^j \sim Q(\hat{v}_2^j|v_2^j)} L(\hat{v}_1^j, \hat{v}_2^j) p(\delta_l^2) \tilde{p}(\delta_2^2) d\delta_l^2 d\delta_2^2$$ (7)

where $A_1^j$ and $A_2^j$ have the same definitions as the symbols in Eq.(6). $p(\delta_l^2)$ and $q(\delta_2^2)$ denote the distribution of edge perturbations in $A_1^j$ and $A_2^j$ respectively.

As discussed earlier, each dimension $v_{1,k}^j$ in the representation vector $v_1^j$ denotes the existence of edge $(v_{1,k}^j, v_{2,k}^j) \in E^1 \in G^1$. We treat each $v_l^j$ as a one-dimensional sequence $v_{1,1}^j, \ldots, v_{1,N_1}^j$, and use one LSTM as the probabilistic model to learn a joint probability $P(\hat{v}_l^j|v_l^j)$ among all dimensions, which is factorized into a product of conditional distributions:

$$P(\hat{v}_l^j|v_l^j) = \prod_{k=1}^{N_1} p(\hat{v}_{1,k}^j|v_{1,k}^j, \ldots, v_{1,(k-1)}^j, v_l^j)$$ (8)

Based on adversarial nodes $v_1^j$, the LSTM takes the hidden state w.r.t. learnt adversarial-free dimensions $[v_{1,1}^j, \ldots, v_{1,(k-1)}^j]$ as input to estimate current adversarial-free dimensions $v_{1,k}^j$ with the conditional probability $p(\hat{v}_{1,k}^j|v_{1,k}^j, \ldots, v_{1,(k-1)}^j, v_l^j)$. Similarly, we employ another LSTM to learn a joint probability $Q(\hat{v}_2^j|v_2^j)$ and neutralize adversarial nodes $v_2^j$ in $G^2$.

The following theorems exhibit the defense loss $L_D$ on adversarial-free nodes in Eq.(7) is equivalent to the original alignment loss $L$ on clean nodes in Eq.(1), if satisfied with some conditions. In addition, they exhibit the existence of an optimal distribution for the APE model to reach a lower bound.

**Theorem 5.1.** If $P(\hat{v}_1^j|v_1^j) = \tau(v_1^j - v_1^j)$ and $Q(\hat{v}_2^j|v_2^j) = \tau(v_2^j - v_2^j)$ for $\forall(v_1^j, v_2^j) \in D$, where $\tau(\cdot)$ is the Dirac delta function, then $L_D$ in Eq.(7) is equivalent to $L$ in Eq.(1).

**Theorem 5.2.** Assuming that $(\delta_1^1, \delta_2^2) = \argmin_{\delta_1^1 \in A_1^j, \delta_2^2 \in A_2^j} \mathcal{L}(v_1^j + \delta_1^1, v_2^j + \delta_2^2)$, if $P(\hat{v}_1^j|v_1^j) = \tau(v_1^j - v_1^j - \delta_1^1)$ and $Q(\hat{v}_2^j|v_2^j) = \tau(v_2^j - v_2^j - \delta_2^2)$ for $\forall(v_1^j, v_2^j) \in D$, then $L_D$ in Eq.(7) achieves its lower bound.

Proof. Please refer to Appendix A.1 for detailed proof of the above two theorems.

**Lemma 5.3.** If $L_D$ in Eq.(7) achieves its lower bound, adversarial perturbation exists only if $\delta_1^1 \notin A_1^j$ and $\delta_2^2 \notin A_2^j$.

5.2 Dirac Delta Approximation

The above theoretical analysis offers a great opportunity to integrate DDA techniques into the LSTM models to make $v_1^j$ (or $v_2^j$) close to $V_1^j$ (or $V_2^j$) as much as possible, and cause $L_D$ on adversarial-free nodes to be identical to $L$ on clean nodes.

In mathematics, the Dirac delta function is a distribution to model the density of an idealized point mass or point charge as a function equal to zero everywhere except for zero and whose integral over the entire real line is equal to one [16]. In practice, a Dirac delta function can be approximated using a commonly used method developed by Zahedi and Tornberg [66].

$$r(x) = \frac{\exp(-yx)}{(1 + \exp(-yx))^2}$$ (9)

where $x$ is a scalar variable and $y$ is a scalar constant. The larger $y$ is, the more accurate the approximation will be. It can be extended to a vector $x = [x_1, \ldots, x_n]$ with the form of $r(x) = \prod_{i=1}^n r(x_i)$.

In this paper, we utilize the DDA method to approximate the generative probability of adversarial-free nodes. Here, we set $y = 4$ to make the maximum of $r(x)$ equal to 1, i.e., $r(x) = 1$ when $x = 0$.

Next, we integrate DDA into $L_D$ to generate a new defense loss $L_{D'}$ for further improving the performance of the APE model: (1) constraint how close $v_1^j$ can deviate from $v_1^j$, i.e., $P(\hat{v}_1^j|v_1^j) = \tau(v_1^j - v_1^j)$ when $v_1^j = v_1^j$ but $P(\hat{v}_1^j|v_1^j) = \tau(v_1^j - v_1^j) = 0$ if $v_1^j \neq v_1^j$; (2) Especially, the vector-based DDA used in this paper makes $P(\hat{v}_1^j|v_1^j) = 1$ only if $v_1^j$ and $v_1^j$ share all neighbors; and (2) restrain $L_D$ to be identical to $L$ without adversarial perturbations.
\[ L_D' = \mathbb{E}_{(v^1_i, v^2_j) \in D} \int_{\delta^1} \int_{\delta^2} \mathbb{E}_{v^1_i - \delta^1, v^2_j - \delta^2} \mathbb{E}_{v^1_j - \delta^1, v^2_j - \delta^2} \rho(\delta^1) \rho(\delta^2) d\delta^1 d\delta^2 \]

After the adversarial nodes \( v^1_i \) and \( v^2_j \) are neutralized to adversarial-free nodes \( \hat{v}^1_i \) and \( \hat{v}^2_j \) by our APE model, \( \hat{v}^1_i \) and \( \hat{v}^2_j \) can be fed into the trained model \( M \) to produce the robust network alignment results without the model retraining.

## 6 Experimental Evaluation

In this section, we perform a set of extensive experiments to evaluate the robustness of our RNA model for network alignment on three groups of datasets: autonomous systems (AS) [1], social networks (SNS) [72], and DBLP coauthor networks [2], as shown in Table 1. Autonomous system is a collection of connected Internet Protocol networks and routers under the control of network operators. Last.fm is a music-oriented online social network that provides a radio streaming service. LiveJournal is an online social platform where users can keep a blog, journal or diary. DBLP is a computer science bibliography website that offers an online search platform where users can keep a blog, journal or diary. DBLP is a computer science bibliography website that offers an online search platform where users can keep a blog, journal or diary.

### Attack Baseline.

**Random Attack** randomly adds and removes edges to generate perturbed graphs. **Meta-Self** [105] is a poisoning attack model for node classification by using gradients to solve the bilevel optimization problem. **GF-Attack** [8] attacks general learning methods by devising new loss and approximating the spectrum. **CD-Attack** [40] hides nodes in the community by attacking the graph autoencoder model. **LowBlow** [20] is a general low-rank adversarial attack model which is able to affect the performance of various graph learning tasks. **GMA** [73] is the first to conduct the adversarial attacks on graph matching (i.e., network alignment) by estimating and maximizing the densities of nodes to be attacked, for pushing them to dense regions in two graphs to generate imperceptible and effective attacks.

### Network Alignment Algorithms.

**SNNA** [39] is an adversarial learning model to solve the weakly-supervised identity alignment problem by incorporating available annotations as the learning guidance. **CrossMNA** [13] is a cross-network embedding based supervised network alignment method by learning inter/intra-embedding vectors for each node and by computing pairwise node similarity scores across networks. Deep graph matching consensus (DGMC) [24] is a supervised graph matching method which aims to reach a data-driven neighborhood consensus between matched node pairs.

### Defense Baselines.

To our best knowledge, this work is the first to deal with the robustness analysis of network alignment against adversarial attacks. We compare our RNA model with three state-of-the-art graph perturbation elimination models: **GCN-Jaccard** [58] eliminates edges that connect nodes with Jaccard similarity of features smaller than a threshold \( r \). Here we use structural features of nodes to calculate the Jaccard similarity. **GCN-SVD** [20] learns a low-rank approximation of the graph to resist high-rank perturbations. Both GCN-Jaccard and GCN-SVD are general perturbation elimination models irrelevant to specific graph learning tasks and architectures. **Pro-GNN** [34] jointly learns a clean graph and a robust graph neural network model from the perturbed graph guided by the low rank and sparsity properties. Notice that it originally targets at defending node classification. In order to make a fair comparison, we change the classification loss \( L_{GNN} \) in the original paper as the alignment loss. We will verify the robustness of three perturbation elimination models by feeding neutralized graphs into the above three network alignment algorithms.

### Variants of our model.

We evaluate five variants of our method to show the strengths of different components. Attack variants: **PGD** only utilizes the basic PGD model [43] to produce adversarial attacks; **SSPGD** uses our proposed attack signal scaling (ASS) plus the PGD model to generate effective attacks. Defense variants: **RNA-A** only employs the basic adversarial perturbation elimination (APE) model (without the ASS and Dirac delta approximation (DDA) models) to neutralize adversarial attacks; **RNA-AD** uses the APE model plus the DDA model (without adversarial attacks as the supervision). **RNA** well eliminates adversarial perturbations with the full support of the APE, DDA, and ASS components.

### Evaluation Metrics.

We use two popular measures in network alignment to evaluate the attack and defense quality: **Accuracy** [9, 67, 69] and **Precision@K** [13, 76]. A larger **Mismatching Rate** (i.e., \( 1 - \text{Accuracy} \) on test data) or a smaller **Precision@K** indicates a better attack, while a higher **Accuracy** or **Precision@K** presents a better defense. \( K \) is fixed to 30 in all tests.

### Experimental Settings.

Our experiments were conducted on a compute server running on Red Hat Enterprise Linux 7.2 with 2 CPUs of Intel Xeon E5-2650 v4 (at 2.66 GHz) and 8 GPUs of NVIDIA GeForce GTX 2080 Ti (with 11GB of GDDR6 on a 352-bit memory bus and memory bandwidth in the neighborhood of 620GB/s), 256GB of RAM, and 1TB of HDD. Overall, our experiments took about 8 days in a shared resource setting. We expect that a consumer-grade single-GPU machine (e.g., with a 1080 Ti GPU) could complete our full set of experiments in around 12-15 days, if its full resources were dedicated.

In our current implementation, the LSTM models in Eq.(10) are implemented as three-layer perceptrons (input-hidden-output). The number of neurons in the hidden layer is set to 500. The model uses a mini-batch of size 500. The learning rate is equal to 0.001. \( \epsilon \) in Algorithm 1 specifies the budget of allowed perturbed edges for each attacked node. Thus, \( \epsilon \) should be a positive integer. In addition, most of real-world graph datasets, e.g., all datasets used in this paper are extremely sparse, i.e., the average node degree of most datasets in this paper is 2 ~ 5. Thus, even if \( \epsilon \) is very small, say 1 or 2, a large number of edges will be modified, which results in noticeable perturbations. For example, AS v1 contains 10,900 nodes.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>AS</th>
<th>SNS</th>
<th>DBLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph</td>
<td>vs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#Nodes</td>
<td>10,900</td>
<td>11,113</td>
<td>5,682</td>
</tr>
<tr>
<td>#Edges</td>
<td>31,180</td>
<td>31,434</td>
<td>23,393</td>
</tr>
<tr>
<td>#Matched Nodes</td>
<td>6,462</td>
<td>2,138</td>
<td>4,000</td>
</tr>
</tbody>
</table>

*Table 1: Statistics of the Datasets*
After that, we stop to attack the rest of nodes. Unless otherwise explicitly stated, we used the following default parameter settings in the experiments. The noise budget $\epsilon$ in adversarial attacks is set to 2. The number limit of perturbed edges in entire graphs is set to 5%. The attack signal threshold $\eta$ to 2. The number limit of perturbed edges in entire graphs is fixed to 5% in these experiments. It is observed that among eight attack methods, no matter how strong the attacks are, our proposed SSPGD attack method achieves the highest mismatching rates on perturbed graphs in most experiments, showing the effectiveness of SSPGD to the adversarial attacks. Compared to the network alignment results under other attack models, SSPGD, on average, achieves 17.9%, 21.2%, and 28.8% improvement of mismatching rates on AS, SNS, and DBLP respectively. The promising performance of SSPGD with all three network alignment models implies that SSPGD has great potential as a general attack solution to other network alignment methods, which is desirable in practice.

Figures 2 and 3 show the network alignment quality under eight attack models by varying the ratios of perturbed edges from 2% to 25%. It is obvious that the attacking performance improves for each attacker with an increase in the number of perturbed edges. This phenomenon indicates that current deep network alignment algorithms are very sensitive to adversarial attacks. SSPGD achieves the lowest Precision values ($< 0.524$), which are much better than other seven methods in most tests. Especially, when the perturbation ratio is larger than 10%, the precision values drop quickly.

### Table 2: Attack performance: Mismatching rate (%) with 5% perturbed edges

<table>
<thead>
<tr>
<th>Attack Model</th>
<th>AS</th>
<th>SNS</th>
<th>DBLP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SNNA</td>
<td>CrossMNA</td>
<td>DGMC</td>
</tr>
<tr>
<td>Clean</td>
<td>53.9</td>
<td>46.6</td>
<td>34.7</td>
</tr>
<tr>
<td>Random</td>
<td>57.5</td>
<td>49.9</td>
<td>37.6</td>
</tr>
<tr>
<td>Meta-Self</td>
<td>63.1</td>
<td>55.1</td>
<td>45.0</td>
</tr>
<tr>
<td>GF-Attack</td>
<td>57.9</td>
<td>53.7</td>
<td>39.5</td>
</tr>
<tr>
<td>CD-ATTACK</td>
<td>59.0</td>
<td>51.7</td>
<td>42.7</td>
</tr>
<tr>
<td>LowBlow</td>
<td>58.2</td>
<td>53.2</td>
<td>41.1</td>
</tr>
<tr>
<td>GMA</td>
<td>64.2</td>
<td>62.9</td>
<td>54.9</td>
</tr>
<tr>
<td>SSPGD</td>
<td>66.6</td>
<td>64.2</td>
<td>53.8</td>
</tr>
</tbody>
</table>

7. https://github.com/ChuXiaokai/CrossMNA
We first use the above three attack models to generate perturbed graphs. In order to verify the robustness to the adversarial poisoning attacks with different levels of noisy data (i.e., the parameters of two attack models in the authors’ implementation, RNA achieves 15.2%, 20.3%, 8.9% Precision boost on these perturbation elimination models are input to three network alignment algorithms, RNA achieves 15.2%, 20.3%, 8.9% Precision boost on these perturbation elimination models. Compared to the network alignment results by all other elimination methods, the alignment results by our proposed RNA ensure informative attack signal propagation in adversarial attacks.

6.2 Defense performance

Figures 4–9 present the network alignment results with the protection of four graph perturbation elimination models on the datasets: AS and SNS. In order to verify the robustness to the adversarial attacks, we evaluate the performance of different alignment methods under three adversarial attack models. For Random Attack, we add and remove the edges to two groups of datasets for performing poisoning attacks with different levels of noisy data (i.e., the ratio of noise edges to original clean edges), say 5% (i.e., 0.05), by randomly adding and removing half edges (e.g., 2.5%) respectively. We perform the network alignment test on original networks with zero noise level and modified ones with different noise levels. In addition, we utilize the LowBlow and GMA attack models to validate the model robustness of different methods. We use the default parameters of two attack models in the authors’ implementation. We first use the above three attack models to generate perturbed graphs. And then they are fed into four perturbation elimination models of GCN-Jaccard, GCN-SVD, Pro-GNN, and our RNA method to eliminate the perturbations. The neutralized graphs generated by these perturbation elimination models are input to three network alignment algorithms for outputting robust alignment results.

We have observed from Figures 4–9 that among four perturbation elimination methods, the alignment results by our proposed RNA on both original and noisy networks achieve the best quality in most experiments, demonstrating the effectiveness and robustness of RNA. Compared to the network alignment results by all other algorithms, RNA achieves 15.2%, 20.3%, 8.9% Precision boost on average by using SNNA, CrossMNA, and DGMC as network alignment methods respectively. These results illustrate that both attack signal scaling and adversarial perturbation elimination are important to solve the robust network alignment problem. The former ensures informative attack signal propagation in adversarial attacks and thus helps identify the really weak features in the graphs for better training adversarial perturbation elimination. On the other
Table 3: Attack: Mismatching rate (%) of SSPGD variants with 5% perturbed edges

<table>
<thead>
<tr>
<th>Alignment Method</th>
<th>AS</th>
<th>SNS</th>
<th>DBLP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SNNA</td>
<td>CrossMNA</td>
<td>DGMC</td>
</tr>
<tr>
<td>PGD</td>
<td>60.5</td>
<td>57.1</td>
<td>48.4</td>
</tr>
<tr>
<td>SSPGD</td>
<td><strong>66.6</strong></td>
<td><strong>64.2</strong></td>
<td><strong>53.8</strong></td>
</tr>
</tbody>
</table>

Table 4: Defense: Precision (%) of RNA variants with 5% perturbed edges

<table>
<thead>
<tr>
<th>Alignment Method</th>
<th>AS</th>
<th>SNS</th>
<th>DBLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNA</td>
<td>35.5</td>
<td>40.8</td>
<td>54.1</td>
</tr>
<tr>
<td>RNA-AD</td>
<td>41.7</td>
<td>45.6</td>
<td>58.2</td>
</tr>
<tr>
<td>RNA</td>
<td><strong>45.4</strong></td>
<td><strong>50.8</strong></td>
<td><strong>62.4</strong></td>
</tr>
</tbody>
</table>

Figure 10: Precision of DGMC with varying parameters

(a) Noise Budget $\epsilon$ in Attack  (b) Signal Threshold $\eta$  (c) Scalar $y$ in Dirac Delta

6.3 Ablation Study

Table 3 presents the mismatching rates of network alignment on three groups of datasets with two variants of our attack model. We have observed that our SSPGD achieves the highest mismatching rates ($> 53.8\%$) on AS, ($> 58.7\%$) over SNS, and ($> 77.4\%$) on DBLP, which are obviously better than PGD. PGD does not utilize our attack signal scaling (ASS) method to solve the gradient vanishing issues and show a fake sense of attack effectiveness. Table 4 reports the precision scores of network alignment with three variants of our adversarial perturbation elimination (APE) model. RNA achieves the best performance in all experiments, while RNA-AD outperforms RNA-A in most experiments. A reasonable explanation is that the ASS model can help generate effective adversarial attacks, which assists our APE model to know what real attacks look like and learn how to combat with them. The Dirac delta approximation (DDA) method ensures the APE model is able to make adversarial-free networks be close to original clean ones as much as possible.

6.4 Parameter Analysis

Figure 10 (a) measures the performance effect of different noise budgets $\epsilon$ for robust network alignment by varying $\epsilon$ from 1 to 6. We have observed when increasing $\epsilon$, the performance scores under three attack models initially keep stable or slightly decreasing but decrease substantially after a certain threshold. This shows it is possible to train a robust perturbation elimination model by utilizing the guidance of adversarial attacks under an appropriate $\epsilon$. However, too many adversarial perturbations can completely destroy the learning process of perturbation elimination. The noise with large $\epsilon$ is easily noticeable by users and thus we suggest generating a robust perturbation elimination solution for network alignment with $\epsilon = 2$ or 3. Also, it seems that the denser the network is, the larger the optimal $\epsilon$ should be. A reasonable explanation is that denser networks need more edge perturbations for each node to change the relative order of the similarity scores between nodes and thus modify alignment results.

Figure 10 (b) shows the influence of the signal threshold $\eta$ in adversarial attacks over two groups of datasets. It is observed that the performance curves initially raises when $\eta$ increases. Intuitively, this can help alleviate the vanishing gradient issue in gradient-based adversarial attacks. The effective adversarial attacks can help the training of the adversarial perturbation elimination model focus on the elimination of the perturbations on the weak edges, and thus improve the alignment robustness. Later on, the performance curves keep decreasing when $\eta$ continuously increases. A rational guess is that a large $\eta$ will make many negative nodes $v_k^2 \neq v_j^2$ have large gradients and thus influence the PGD computation in Eq. (2). However, the success of attacks mainly depends on the single $v_k^2$ that is most similar to $v_j^2$, instead of all negative nodes.

Figure 10 (c) exhibits the impact of the scalar constant $y$ in Dirac delta approximation. The experimental results are consistent with the discussion in Eq. (9), i.e., the larger $y$ is, the more accurate the approximation will be. This approximation is a commonly used in practice for the delta function in some existing works. We set $y = 4$ to approximate the generative probability of adversarial-free nodes. The performance curves keep relatively stable when $y$ continuously increases to 5.

7 RELATED WORK

Adversarial defenses on graphs. Recent studies have presented that graph learning models, especially deep learning models, are highly sensitive to adversarial attacks, i.e., carefully designed small deliberate perturbations in graph structure and attributes can cause the prediction failures of the models. Only recently, researchers have started to develop adversarial defense approaches to improve the robustness of graph mining models against adversarial attacks in node classification [19, 20, 23, 34, 55, 62, 71, 74, 78, 103], network embedding [15], link prediction [77], malware detection [29], spammer detection [17], fraud detection [7, 70], and influence maximization [42]. Certifiable robustness techniques aim to design robustness certificates to measure the safety of individual nodes under adversarial perturbation. Training learning models jointly with...
these certificates can lead to a rigorous safety guarantee of more nodes in various tasks, include node classification [5, 6, 106, 107], graph classification [32], and community detection [31].

Network alignment. Graph data analysis has attracted active research in the last decade [4, 10–12, 26, 27, 36, 37, 46, 47, 51–53, 59, 60, 73, 80–102]. It is also well known as graph matching and has been a heated topic in recent years [13, 41, 72, 76]. It is also well known as graph matching and has been a heated topic in recent years

8 CONCLUSIONS
We have presented a robust network alignment solution. First, we analyze how gradient vanishing causes failures of gradient-based adversarial attacks. Second, we design an attack signal scaling method to ensure informative signal propagation. Finally, we develop an adversarial perturbation elimination model to neutralize adversarial nodes in vulnerable space to adversarial-free nodes in safe area.

REFERENCES
A APPENDIX

A.1 Theoretical Proof

Theorem 5.1. Let nodes $v_j^1$ and $v_k^2$ be the most similar and least similar to $M(v_i^1)^T$ (1 ≤ j, k ≤ N²), i.e., $v_j^1 = \arg\max_{v_j} (M(v_i^1)^T \cdot v_j^1)$ and $v_k^2 = \arg\min_{v_k} (M(v_i^1)^T \cdot v_k^2)$, and $c = (M(v_i^1)^T \cdot v_j^1)$. Also, suppose that $d$ is the minimal norm of node representation vectors in $\mathbb{R}^2$, i.e., $d = \min_v \|\delta v\|_2$ for $\forall v' \in \mathbb{V}$. For a given $0 < \eta < d/2$, if

$$\alpha < \frac{1}{\delta} \log \frac{d \eta}{\eta}, \text{ then } 1 - \sigma(\alpha M(v_i^1)^T \cdot v_j^1) > \eta/\|v_j^1\|_2 \text{ for } \forall v' \in \mathbb{V}^2.$$

Proof. From Theorem 6.2, we have $\|v_k^2\|_2 \leq c$, thus we can convert it to $1 - \exp(-\alpha M(v_i^1)^T \cdot v_j^1) < 1 - \eta/\|v_j^1\|_2$. If we can prove $\eta/\|v_j^1\|_2 < \eta/\|v_k^2\|_2$, then we can satisfy this condition. Thus, we need to solve

$$\exp(\alpha c) < \frac{\|v_k^2\|_2 - \eta}{\eta}.$$

Since $\exp$ is a monotonic increasing function, by solving the above inequality, we have $\alpha < \frac{1}{\delta} \log \frac{\|v_k^2\|_2 - \eta}{\eta}$. As $\|v_k^2\|_2 \geq d$ and $\alpha < \frac{1}{\delta} \log \frac{d \eta}{\eta}$, the above statement is proved.

Notice that $0 < \eta < d/2$. This makes $\frac{d \eta}{\eta} > 1$ and the upper bound of $\alpha$ is positive. Therefore, for any $\alpha < \frac{1}{\delta} \log \frac{d \eta}{\eta}$, we have $1 - \sigma(\alpha M(v_i^1)^T \cdot v_j^1) > \eta/\|v_j^1\|_2$ is satisfied.

Theorem 6.2. If $P(v_i^1 | v_j^1) = \tau(v_i^1 - v_j^1)$ and $Q(v_i^2 | v_j^2) = \tau(v_i^2 - v_j^2)$ for $\forall v_i^1, v_j^2 \in \mathcal{D}$, where $\tau(\cdot)$ is the Dirac delta function, then $\mathcal{L}_D$ in Eq.(7) is equivalent to $\mathcal{L}$ in Eq.(1).

Proof.

$\mathcal{L}_D = E_{(v_i^1, v_j^2)} \int_{\Delta_j^1} \int_{\Delta_j^2} p(\delta_i^j) d\delta_i^j \int_{\Delta_j^2} q(\delta_j^2) d\delta_j^2$

$$\int_{v_j^1} \tau(v_i^1 - v_j^1) dv_j^1 \int_{v_j^2} \tau(v_i^2 - v_j^2) dv_j^2$$

$$= E_{(v_i^1, v_j^2)} \int_{\Delta_j^1} p(\delta_i^j) d\delta_i^j \int_{\Delta_j^2} q(\delta_j^2) d\delta_j^2$$

$$\int_{v_j^1} P(v_i^1 | v_j^1) dv_i^1 \int_{v_j^2} Q(v_i^2 | v_j^2) \mathcal{L}(v_i^1, v_j^2) dv_j^2$$

$$= E_{(v_i^1, v_j^2)} \int_{\Delta_j^1} p(\delta_i^j) d\delta_i^j \int_{\Delta_j^2} q(\delta_j^2) d\delta_j^2$$

$$\int_{v_j^1} \tau(v_i^1 - v_j^1) dv_i^1 \int_{v_j^2} \tau(v_i^2 - v_j^2) \mathcal{L}(v_i^1, v_j^2) dv_j^2$$

(12)

$$\mathcal{L}_D = E_{(v_i^1, v_j^2)} \int_{\Delta_j^1} \int_{\Delta_j^2} p(\delta_i^j) d\delta_i^j \int_{\Delta_j^2} q(\delta_j^2) d\delta_j^2$$

$$\int_{v_j^1} \tau(v_i^1 - v_j^1) dv_i^1 \int_{v_j^2} \tau(v_i^2 - v_j^2) \mathcal{L}(v_i^1, v_j^2) dv_j^2$$

(13)

The equality is satisfied when $P(v_i^1 | v_j^1) = \tau(v_i^1 - v_j^1 - \delta_i^j)$ and $Q(v_i^2 | v_j^2) = \tau(v_i^2 - v_j^2 - \delta_j^2)$. According to the sifting property of the integral of Dirac delta function, we apply the sifting operation twice and have

$$\mathcal{L}_D = E_{(v_i^1, v_j^2)} \int_{\Delta_j^1} \int_{\Delta_j^2} p(\delta_i^j) d\delta_i^j \int_{\Delta_j^2} q(\delta_j^2) d\delta_j^2$$

$$\int_{v_j^1} \tau(v_i^1 - v_j^1) dv_i^1 \int_{v_j^2} \tau(v_i^2 - v_j^2) \mathcal{L}(v_i^1, v_j^2) dv_j^2$$

(11)