Paper: On the Power of GNA using DPS

Created	@September 12, 2023 10:45 AM
Type	Reading
Materials	https://arxiv.org/pdf/2201.10945.pdf
Reviewed	☐

3. METHODOLOGY

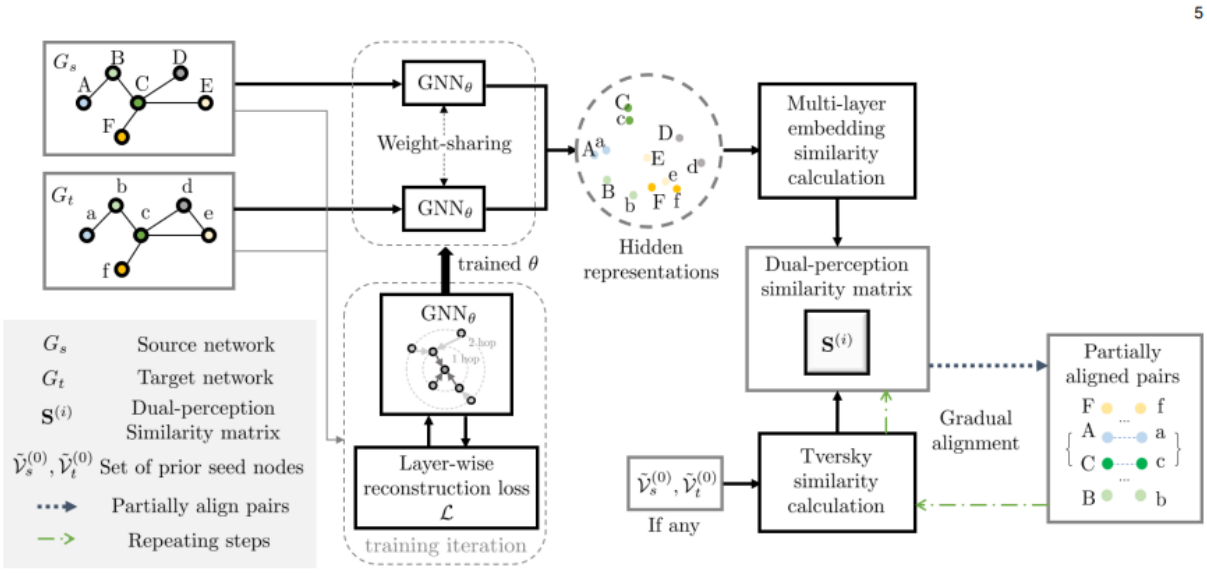

Fig. 3: The schematic overview of our Grad-Align method.

Notation	Description
G_s	Source network
$\mathcal{V}_s$	Set of nodes in G_s
$\mathcal{E}_s$	Set of edges in G_s
$\mathcal{X}_s$	Set of attributes of nodes in $\mathcal{V}_s$
G_t	Target network
$\mathcal{V}_t$	Set of nodes in G_t
$\mathcal{E}_t$	Set of edges in G_t
$\mathcal{X}_t$	Set of attributes of nodes in $\mathcal{V}_t$
n_s	Number of nodes in G_s
n_t	Number of nodes in G_t
$\pi^{(i)}$	One-to-one node mapping at the i-th iteration
M	Total number of ground truth node pairs
$\tilde{\mathcal{V}}_s^{(i)}$	Set of seed nodes in G_s up to the i-th iteration
$\tilde{\mathcal{V}}_t^{(i)}$	Set of seed nodes in G_t up to the i-th iteration
$\hat{\mathcal{V}}_s^{(i)}$	Set of newly aligned nodes in G_s at the i-th iteration
$\hat{\mathcal{V}}_t^{(i)}$	Set of newly aligned nodes in G_t at the i-th iteration
$\tilde{\mathcal{V}}_s^{(0)}$	Set of prior seed nodes in G_s
$\tilde{\mathcal{V}}_t^{(0)}$	Set of prior seed nodes in G_t
$\mathbf{H}_s^{(l)}$	Hidden representation in G_s at the l-th GNN layer
$\mathbf{H}_t^{(l)}$	Hidden representation in G_t at the l-th GNN layer
$\mathbf{S}_{emb}$	Multi-layer embedding similarity matrix
$\mathbf{S}_{Tve}^{(i)}$	Tversky similarity matrix at the i-th iteration
$\mathbf{S}^{(i)}$	Dual-perception similarity matrix at the i-th iteration

TABLE 1: Summary of notations.

Definition 3.1 (NA). *Given two networks* $G_s = (\mathcal{V}_s, \mathcal{E}_s, \mathcal{X}_s)$ *and* $G_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{X}_t)$*, NA aims to find one-to-one mapping* $\pi : \mathcal{V}_s \to \mathcal{V}_t$*, where* $\pi(u) = v$ *and* $\pi^{-1}(v) = u$ *for* $u \in \mathcal{V}_s$ *and* $v \in \mathcal{V}_t$.

Trong 2 cái đồ thị là G_source và G_target thì Network Alignment sẽ có xu hướng tìm để nối 2 cặp node giữa 2 đồ thị trong đó sử dụng hàm pi với đầu vào là node của 1 đồ thị đầu ra là node của đồ thị còn lại.

Grad-Align is basically composed of 3 main phases:

1) the multi-layer embedding similarity calculation.

(Phase 1) We first focus on calculating the *multi-layer* embedding similarity matrix via L-layer GNN. Let $\mathbf{H}_s^{(l)} \in \mathbb{R}^{n_s \times h}$ and $\mathbf{H}_t^{(l)} \in \mathbb{R}^{n_t \times h}$ denote the hidden representation in G_s and G_t, respectively, at the l-th GNN layer where $l \in \{1, \cdots, L\}$ and h is the dimension of each representation vector. As illustrated in Fig. 3, in the feed-forward process of GNN, we use the weight-sharing technique in order to consistently generate $\mathbf{H}_s^{(l)}$ and $\mathbf{H}_t^{(l)}$ at each layer. We then train the generated GNN model parameters via a *layer-wise reconstruction loss* to make each node representation more distinguishable by exploiting the nodes' proximities up to the l-th order (which will be specified in Section 4.1). Using the hidden representations $\mathbf{H}_s^{(l)}$ and $\mathbf{H}_t^{(l)}$ at each layer through training iterations, we are capable of computing the multi-layer embedding similarity matrix $\mathbf{S}_{emb} \in \mathbb{R}^{n_s \times n_t}$, which is expressed as

$$\mathbf{S}_{emb} = \sum_l \mathbf{H}_s^{(l)} \mathbf{H}_t^{(l)\top}, \quad (1)$$

where the superscript $\top$ denotes the transpose of a matrix and each entry in $\mathbf{S}_{emb}$ represents the similarity of pairwise node embedding vectors in the two networks G_s and G_t.

Giai đoạn này thì nó sẽ tạo ra 1 mạng neuron đồ thị (GNN) gồm L lớp (Layer). Với mỗi lớp thứ l thuộc tập {1,…,L} thì nó sẽ tạo cho mỗi đồ thị 1 biểu diễn ẩn (hidden representation) thuộc tập R^(n * h) với n là số node của đồ thị và h là số chiều của vector biểu diễn.

Tiếp theo huấn luyện các thang đo hay mình hiểu thì là trọng số dựa vào 1 cái gọi là mất mát tái thiết theo lớp (layer-wise recontruction loss) nhằm bộc lộ rõ hơn các node bằng việc khám phá độ gần giữa chúng ở từng lớp.

Từ đó ta sẽ có các biểu diễn ẩn sau khi đã train qua model GNN để tính được ma trận tương quan ẩn nhiều lớp qua công thức trên.

2) the Tversky similarity calculation.

Definition 3.2. *(ACN). Given a node pair (u, v), if $x \in \mathcal{N}_{G_s,u}$, $y \in \mathcal{N}_{G_t,v}$, and $\pi^{(i)}(x) = y$ (i.e., (x, y) is the already aligned node pair), then the node pair (x, y) belongs an ACN of (u, v) across two networks.*

ACN (aligned cross-network neighbor-pair): có một cặp nút u,v lần lượt thuộc 2 network A, B, tồn tại một nút x thuộc tập hàng xóm của nút u và một nút y thuộc

tập hàng xóm của nút v, trong trường hợp x và y đã được aligned (mapping_func(x) tại i = y) => (x,y) được coi là 1 ACN của cặp nút (u,v)

Ở phase 2 ta sẽ có 1 tập hợp với ký hiệu: $\mathcal{N}_{G_s,u}$

Trong đó thì Gs là đồ thị nguồn, u là 1 node trong đồ thị nguồn và tập này đại diện so các hàng xóm của node u trong đồ thị nguồn Gs. Trong khi lặp lại thực hiện tính toán độ tương quan Tversky thì mình định nghĩa hàm pi^(i) là ánh xạ của nó tại vòng lặp thứ i. Qua mỗi vòng lặp tính số ACN của mỗi cặp node (u,v) tương ứng dựa trên hàm ánh xạ node 1-1 pi^(i). Tính ma trận tương quan Tversky lặp đi lặp lại đến khi cân bằng sự khác biệt giữa hai tập hàng xóm. Cuối cùng tính ma trận tương quan hai góc độ (dual-perception similarity matrix) ở vòng lặp thứ i bằng cách nhân ma trận từng phần tử với nhau giữa ma trận tương quan Tversky (Tversky similarity matrix) ở vòng lặp thứ i và ma trận tương quan ẩn nhiều lớp.

$$\mathbf{S}^{(i)} = \mathbf{S}_{emb} \odot \mathbf{S}_{Tve}^{(i)},$$

$\mathbf{S}^{(i)} \in \mathbb{R}^{n_s \times n_t}$ $\mathbf{S}_{Tve}^{(i)} \in \mathbb{R}^{n_s \times n_t}$ $\mathbf{S}_{emb} \in \mathbb{R}^{n_s \times n_t}$,

3) the gradual node matching.

(Phase 3) We explain how to gradually match node pairs using our dual-perception similarity $\mathbf{S}^{(i)}$ in (2) for each iteration. Since $\mathbf{S}_{emb}$ does not change over iterations, we focus only on how to update the Tversky similarity $\mathbf{S}_{Tve}^{(i)}$. From the fact that the NA problem is often solved under the supervision of some observed anchor nodes [1], [2], [7], we suppose that there are two sets of *prior seed nodes* (also known as anchor nodes) in two different networks, denoted

Vì ma trận tương quan ẩn nhiều lớp nó cố định qua mỗi vòng lặp nên ta chỉ tập trung tính ma trận tương quan Tversky, trong đó lấy 2 tập hợp là tập các node quan trọng (anchor node) của 2 đồ thị qua mỗi vòng lặp. Khi đó ma trận tương quan Tversky của vòng lặp tiếp theo sẽ được tính cùng với hàm ánh xạ node pi^(i). Từ việc lặp lại ta sẽ có được các cặp node đã được căn chỉnh tạo bởi 2

by $\tilde{\mathcal{V}}_s^{(0)}$ and $\tilde{\mathcal{V}}_t^{(0)}$ in source and target networks, respectively, whose links connecting them correspond to the ground truth prior anchor information for NA. These two sets are utilized to *initially* calculate $\mathbf{S}_{Tve}^{(1)}$ along with the node mapping function $\pi^{(0)}$.[3] By iteratively updating $\mathbf{S}_{Tve}^{(i)}$ based on the information of newly aligned node pairs, Grad-Align makes full use of node pairs exhibiting strong consistency as well as prior seed node pairs (i.e., pairs that connect $\tilde{\mathcal{V}}_s^{(0)}$ and $\tilde{\mathcal{V}}_t^{(0)}$). Next, we explain our gradual alignment process. Let $\tilde{\mathcal{V}}_s^{(i)}$ and $\tilde{\mathcal{V}}_t^{(i)}$ denote the set of aligned nodes in G_s and G_t, respectively, up to the i-th iteration. Then, it follows that $\tilde{\mathcal{V}}_*^{(i)} = \tilde{\mathcal{V}}_*^{(i-1)} \cup \hat{\mathcal{V}}_*^{(i)}$, where the subscript * represents s and t for source and target networks, respectively, and $\hat{\mathcal{V}}_*^{(i)}$ is the set of *newly* aligned nodes in each network at the i-th iteration. By iteratively calculating the dual-perception similarity matrix $\mathbf{S}^{(i)}$ in (2), we discover $|\hat{\mathcal{V}}_s^{(i)}| = |\hat{\mathcal{V}}_t^{(i)}| = N$ cross-network node pairs at each iteration, where $|\cdot|$ is the cardinality of the set, and $N > 0$ is a positive integer. To this end, similarly as in [1], [19], we employ a ranking-based selection strategy that selects the top-N elements in the matrix $\mathbf{S}^{(i)}$ for each iteration (which will be specified in Section 4.1). Then, since the node mapping $\pi^{(i)}$ is updated to $\pi^{(i+1)}$ after N node pairs are newly found, the matrix $\mathbf{S}_{Tve}^{(i+1)}$ is recalculated accordingly in order to update $\pi^{(i+2)}$. The process is repeated until all node correspondences (i.e., M node pairs) are found. Specifically, the iteration ends when i becomes $\lceil \frac{M}{N} \rceil + 1$, where $\lceil \cdot \rceil$ is the ceiling operator.[4]

tập node anchor. Đồng thời các tập mới sẽ được tính theo công thức qua các vòng lặp.

$$\tilde{\mathcal{V}}_*^{(i)} = \tilde{\mathcal{V}}_*^{(i-1)} \cup \hat{\mathcal{V}}_*^{(i)}$$

Với tập ở dưới là tập các node đã được căn chỉnh ở vòng lặp thứ i:

$$\hat{\mathcal{V}}_*^{(i)}$$

Sau khi lặp lại việc tính ma trận tương quan hai góc độ ta sẽ có tập các node đã được căn chỉnh ở vòng lặp thứ i của cả hai đồ thị sẽ có số phần tử là bằng nhau và bằng 1 giá trị N dương. Tiếp theo mình xếp hạng để chọn ra tóp N phần tử trong ma trận tương quan 2 góc độ ở mỗi vòng lặp.Khi hàm pi^i được cập nhất lên pi^(i + 1) sau N cặp node mới thì ma trận tương quan Tversky ở vòng lặp i + 1 sẽ được tính lại để cập nhật pi^(i + 2). Vòng lặp lại (M/N) + 1 lấy ceiling.

Algorithm 1 : Grad-Align

Input: $G_s, G_t, \tilde{\mathcal{V}}_s^{(0)}, \tilde{\mathcal{V}}_t^{(0)}$
Output: $\pi^{(\lceil \frac{M}{N} \rceil + 1)}$

Initialization: $\theta \leftarrow$ random initialization; $i \leftarrow 1$
/* Calculation of $\mathbf{S}_{emb}$ */
while not converged **do**
  $\mathbf{H}_s^{(1)}, \mathbf{H}_s^{(2)}, ..., \mathbf{H}_s^{(L)} \leftarrow \text{GNN}_\theta(G_s)$
  $\mathbf{H}_t^{(1)}, \mathbf{H}_t^{(2)}, ..., \mathbf{H}_t^{(L)} \leftarrow \text{GNN}_\theta(G_t)$
  $\mathcal{L} \leftarrow$ layer-wise reconstruction loss in (8)
  Update θ by taking one step of gradient descent
end while
$\mathbf{S}_{emb} \leftarrow \sum_{l=1}^{L} \mathbf{H}_s^{(l)} \mathbf{H}_t^{(l)\top}$
/* Gradual matching by updating $\mathbf{S}_{Tve}^{(i)}$ */
if $\tilde{\mathcal{V}}_s^{(0)} = \tilde{\mathcal{V}}_t^{(0)} = \emptyset$ **then**
  $\mathbf{S}^{(i)} \leftarrow \mathbf{S}_{emb}$
else
  Calculate $\mathbf{S}_{Tve}^{(i)}$ using (9)
  $\mathbf{S}^{(i)} \leftarrow \mathbf{S}_{emb} \odot \mathbf{S}_{Tve}^{(i)}$
end if
while $i \leq \lceil \frac{M}{N} \rceil$ **do**
  Find $\hat{\mathcal{V}}_s^{(i)}$ and $\hat{\mathcal{V}}_t^{(i)}$ based on $\mathbf{S}^{(i)}$
  $i \leftarrow i + 1$
  Update mapping $\pi^{(i)}$:
  $\{\pi^{(i)}(u) = v \mid u \in \hat{\mathcal{V}}_s^{(i-1)}, v \in \hat{\mathcal{V}}_t^{(i-1)}\}$
  Update $\mathbf{S}_{Tve}^{(i)}$ using (9)
  $\mathbf{S}^{(i)} \leftarrow \mathbf{S}_{emb} \odot \mathbf{S}_{Tve}^{(i)}$
end while
return $\pi^{(\lceil \frac{M}{N} \rceil + 1)}$

4 PROPOSED GRAD-ALIGN METHOD

4.1 Methodological Details of Grad-Align

Phương pháp hay công thức để tính ma trận tương quan ẩn đa chiều Semb là ta sẽ dùng 1 model GNN. Dạng chuẩn của cơ chế truyền tin vào GNN trong đó ở liên

tục cập nhật biểu diễn của từng node bằng cách tổng hợp biểu diễn của các nút lân cận bằng hai hàm, cụ thể là AGGREGATE và UPDATE.

$$\mathbf{m}_u^l \leftarrow \text{AGGREGATE}_u^l(\{\mathbf{h}_{x_u}^{l-1} | x_u \in \mathcal{N}_u \cup u\}, \mathbf{W}_1^l)$$

where $\mathbf{h}_{x_u}^{l-1} \in \mathbb{R}^{1\times h}$ denotes the h-dimensional latent representation vector of node x_u at the $(l-1)$-th hidden layer; $\mathcal{N}_u$ indicates the set of neighbor nodes of u; $\mathbf{W}_1^l \in \mathbb{R}^{h\times h}$ is the learnable weight matrix at the l-th layer for the AGGREGATE step; and $\mathbf{m}_u^l$ is the aggregated information at the l-th layer. Here, $\mathbf{h}_u^0$ represents the node attribute vector of node u. In the UPDATE step, the latent representation at the next hidden layer is updated by using each node and its aggregated information from AGGREGATE_u^l as follows:

Hàm AGGREGATE ở đây được sử dụng để tổng hợp thông tin đặc điểm của từng node u ở mỗi layer l.

$$\mathbf{h}_u^l \leftarrow \text{UPDATE}_u^l(u, \mathbf{m}_u^l, \mathbf{W}_2^l)$$

where $\mathbf{W}_2^l \in \mathbb{R}^{h^m \times h^l}$ is the learnable weight matrix at the l-th layer for the UPDATE step.[5] The above two functions

Hàm UPDATE sử dụng để cập nhật biểu diễn tiềm ẩn ở hidden layer tiếp theo được cập nhật là các node và thông tin tổng hợp từ hàm AGGREGATE.

Proposition 4.1. *Suppose that two networks G_s and G_t are isomorphic,[6] where any of node pairs (u, v) matched by ground truth node mapping $\pi : \mathcal{V}_s \to \mathcal{V}_t$ have the same node attributes $\mathbf{h}_u^0 = \mathbf{h}_v^0$. Given two GNN models for G_s and G_t, if the weight matrices $\{\mathbf{W}_1^p\}_{p=1,\cdots,L}$ and $\{\mathbf{W}_2^p\}_{p=1,\cdots,L}$ are shared, then it follows that $\mathbf{h}_u^p = \mathbf{h}_v^p$ for all positive integers p, where L indicates the number of GNN layers.*

Proof

Base step: For node $u \in \mathcal{V}_s$, it follows that

$$\mathbf{m}_u^1 \leftarrow \text{AGGREGATE}_u^1(\{\mathbf{h}_x^0 | x \in \mathcal{N}_u \cup u\}, \mathbf{W}_1^1)$$
$$\mathbf{h}_u^1 \leftarrow \text{UPDATE}_u^1(u, \mathbf{m}_u^1, \mathbf{W}_2^1).$$

For node $v \in \mathcal{V}_t$, we have

$$\mathbf{m}_v^1 \leftarrow \text{AGGREGATE}_v^1(\{\mathbf{h}_y^0 | y \in \mathcal{N}_v \cup v\}, \mathbf{W}_1^1)$$
$$\mathbf{h}_v^1 \leftarrow \text{UPDATE}_v^1(v, \mathbf{m}_v^1, \mathbf{W}_2^1).$$

$$\{h_x^0 \,|\, x \in \mathbb{N}_u \cup u\} = \{h_y^0 \,|\, y \in \mathbb{N}_v \cup v\}$$

Sự bằng nhau này xảy ra khi mà hai đồ thị Gs và Gt là đẳng hình và $h_u^0 = h_v^0$

Do đó t sẽ có dựa vào hàm AGGREGATE thì $m_u^1 = m_v^1$

Tiếp đó dùng hàm UPDATE dựa vào kết quả vừa có t nhận được $h_u^1 = h_v^1$

for $p = k$, where $k \in \{1, \cdots, L-1\}$. Then, it is not difficult to show that (7) also holds for $p = k + 1$ using (3) and (4), which completes the proof of this proposition. □

Bây giờ mình sẽ huấn luyện model GNN với các trọng số trong tập theta, đồng thời sử dụng kỹ thuật chia sẻ trọng số giữa những cái GNN với nhau. Từ đó tăng cường sự nhất quán giữa 2 đồ thị với nhau trong cùng 1 không gian tiềm ẩn.

Mình sẽ sử dụng hàm layer-wise reconstruction loss trong việc huấn luyện model GNN

$$\mathcal{L} = \sum_{*\in\{s,t\}} \sum_{l} \left\| \tilde{D}_*^{(l)-\frac{1}{2}} \tilde{A}_*^{(l)} \tilde{D}_*^{(l)-\frac{1}{2}} - H_*^{(l)} H_*^{(l)\top} \right\|_F$$

where the subscript * represents s and t for source and target networks, respectively; $\tilde{A}_*^{(l)} = \sum_{k=1}^{l} \hat{A}_*^k$ where $\hat{A}_* = A_* + I_*$ is the adjacency matrix with self-connections in which I_* is the identity matrix; $\tilde{D}_*^{(l)}$ is a diagonal matrix whose (i,i)-th element is $\tilde{D}_*^{(l)}(i,i) = \sum_j \tilde{A}_*^{(l)}(i,j)$ where $\tilde{A}_*^{(l)}(i,j)$ is the (i,j)-th element of $\tilde{A}_*^{(l)}$; and $\|\cdot\|_F$ is the Frobenius norm of a matrix. By training the GNN model via our layer-wise reconstruction loss in (8), we are capable of precisely capturing the first-order and higher-order neighborhood structures. Note that, unlike the prior study (e.g., [2]), we use the *aggregated* matrix $\tilde{A}_*^{(l)}$ to reconstruct the underlying network from each hidden representation.

$$\|A\|_F \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}$$

Frobenius norm of a matrix

$D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ represents a matrix similarity transformation

$$H_*^{(l)} H_*^{(l)\top}$$

represents node similarity after they have been embedded

Quay trở lại tính toán ma trận tương quan Tversky , ta giả sử

$$n_s \geq n_t$$

cho phép tính độ tương quan Tversky. Cho 1 cặp node cross-network (u, v) ta sẽ tính toán số ACN được chia sẻ giữa 2 tập

$$\mathcal{N}_{G_s,u} \text{ and } \mathcal{N}_{G_t,v}$$

Trong đó mô tả tập hàng xóm trong 1 bước nhảy của u trong Gs và v trong Gt. Sử dụng hàm ánh xạ pi tại vòng lặp thứ i.

Khi đó ta có công thức:

$$\mathbf{S}_{Tve}^{(i)}(u,v) = \frac{|X_u^{(i)} \cap Y_v^{(i)}|}{|X_u^{(i)} \cap Y_v^{(i)}| + \alpha|X_u^{(i)} - Y_v^{(i)}| + \beta|Y_v^{(i)} - X_u^{(i)}|}$$

where

$$\mathcal{T}_u^{(i)} = \left\{\pi^{(i)}(x) \mid x \in \left(\mathcal{N}_{G_s,u} \cap \tilde{\mathcal{V}}_s^{(i)}\right)\right\}$$
$$X_u^{(i)} = \left(\mathcal{N}_{G_s,u} - \tilde{\mathcal{V}}_s^{(i)}\right) \cup \mathcal{T}_u^{(i)}$$
$$Y_v^{(i)} = \mathcal{N}_{G_t,v}.$$

Here, $\tilde{\mathcal{V}}_s^{(i)}$ is the set of *seed nodes* at the i-th iteration of gradual alignment in G_s, which is defined as the union of already aligned node pairs up to the i-th iteration and $\tilde{\mathcal{V}}_s^{(0)}$ (i.e., prior seed nodes in G_s); and $\alpha, \beta > 0$ are the parameters of the Tversky index. In our study, we set $0 < \alpha < 1$ and $\beta = 1$ under the condition $n_s \geq n_t$, where the appropriate value of α will be numerically found in Section 5.5.2. It is worth noting that the case of $\alpha = \beta = 1$ corresponds to the Tanimoto coefficient (also known as the Jaccard index) [12], which can be seen as a symmetric set similarity measure. In particular, we would like to address the importance of the parameter setting in our Tversky similarity.

Remark 1. *The parameter α in (9) plays a crucial role in balancing between $|X_u^{(i)} \cap Y_v^{(i)}|$ (i.e., the number of ACNs of node pair (u, v)) and $|X_u^{(i)} - Y_v^{(i)}|$ (i.e., the cardinality of the set difference of $X_u^{(i)}$ and $Y_v^{(i)}$). More specially, for $n_s \gg n_t$, a number of nodes in G_s remain unaligned even when the alignment process is completed; thus, the term $|X_u^{(i)} \cap Y_v^{(i)}|$ would be far smaller than $|X_u^{(i)} - Y_v^{(i)}|$.[7] In this case, we are capable of more precisely calculating the similarity between node pairs of two networks that are severely imbalanced during the gradual alignment by adjusting the value of α accordingly. By setting $0 < \alpha < 1$, we can mitigate the impact of $|X_u^{(i)} - Y_v^{(i)}|$ so that $|X_u^{(i)} \cap Y_v^{(i)}|$ is relatively influential.*

Giả sử mình gọi cái tập thứ nhất trong phần này là a tập thứ hai là b. Khi đó trong trường hợp số node trong đồ thị S rất lớn so với của đồ thị T khi đó thì tập a sẽ bé hơn rất nhiều so với tập b. Và trong trường hợp đó để ta có thể tính toán độ tương quan giữa các cặp node ở 2 đồ thị khác nhau 1 cách chính xác t cần 1 tham số alpha nữa để điều chỉnh độ chênh lệch đó và alpha sẽ thuộc khoảng (0-1).

Trong khi số ACN tăng lên qua mỗi vòng lặp của quá trình Gradual Alignment, ta cần để tâm tới tốc độ thay đổi của độ tương quan Tversky theo độ tăng đó. Khi đó ta cần cài đặt các trọng số để thể hiện độ hiệu quả của độ tương quan Tversky.

Theorem 4.2. *Given a node pair (u, v), suppose that $|Y_v^{(i)} - X_u^{(i)}| \geq |X_u^{(i)} \cap Y_v^{(i)}|$ for all gradual alignment steps. Then, under the condition $n_s \geq n_t$, the growth rate of our Tversky similarity $\mathbf{S}_{Tve}^{(i)}(u, v)$ using $0 < \alpha < 1$ and $\beta = 1$ with respect to $|X_u^{(i)} \cap Y_v^{(i)}|$ (i.e., the number of ACNs of node pair (u, v) at the i-th iteration) is always higher than that of the Jaccard index using $\alpha = \beta = 1$.*

Lại đặt tập 1 là a, tập 2 là b. Ta có 1 cặp node (u, v) mà trong đó tập a ≥ tập b với mọi bước sắp xếp tuần tự hay Gradual Alignment. Với điều kiện số node của S ≥

số node của T, thì chỉ số độ tương quan Tversky với các tham số alpha thuộc (0-1) và beta = 1 sẽ luôn lớn hơn chỉ số Jaccard với alpha = beta = 1.

Proof. Let $\mathbf{S}^{(i)}_{Jac}(u, v)$ denote the Jaccard index between two nodes $u \in \mathcal{V}_s$ and $v \in \mathcal{V}_t$ in the i-th iteration. For notational convenience, by denoting $a = |X^{(i)}_u - Y^{(i)}_v|$, $b = |Y^{(i)}_v - X^{(i)}_u|$, and $c = |X^{(i)}_u \cap Y^{(i)}_v|$ for $b \geq c \geq 0$, we have

$$\begin{aligned} \mathbf{S}^{(i)}_{Tve}(u, v) &= \frac{c}{\alpha a + b + c} = 1 - \frac{\alpha a + b}{\alpha a + b + c} \\ \mathbf{S}^{(i)}_{Jac}(u, v) &= \frac{c}{a + b + c} = 1 - \frac{a + b}{a + b + c}. \end{aligned} \tag{11}$$

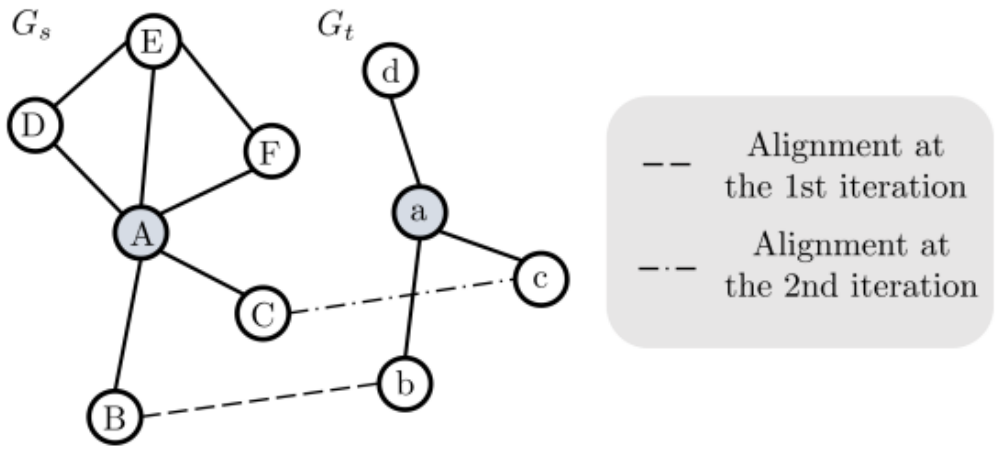

Fig. 4: An example illustrating gradual alignment of two networks G_s and G_t with $n_s = 6$ and $n_t = 4$.

Then, the difference between the growth rates of $\mathbf{S}^{(i)}_{Tve}(u,v)$ and $\mathbf{S}^{(i)}_{Jac}(u,v)$ with respect to c is given by

$$\begin{aligned}\frac{\partial \mathbf{S}^{(i)}_{Jac}(u,v)}{\partial c} - \frac{\partial \mathbf{S}^{(i)}_{Tve}(u,v)}{\partial c} &= \frac{w_1}{(w_1+c)^2} - \frac{w_2}{(w_2+c)^2} \\ &= \frac{(w_1-w_2)(c^2-w_1w_2)}{(w_1+c)^2(w_2+c)^2},\end{aligned} \tag{12}$$

where $w_1 = a + b$ and $w_2 = \alpha a + b$. Here, due to the fact that $w_1 - w_2 = (1-\alpha)a \geq 0$, $w_1 \geq c$, and $w_2 \geq c$, it follows that

$$\frac{\partial \mathbf{S}^{(i)}_{Tve}(u,v)}{\partial c} \geq \frac{\partial \mathbf{S}^{(i)}_{Jac}(u,v)}{\partial c}, \tag{13}$$

which completes the proof of this theorem. □

Example 1. We show how using the Tversky similarity $\mathbf{S}^{(i)}_{Tve}(u,v)$ is beneficial over the Jaccard index $\mathbf{S}^{(i)}_{Jac}(u,v)$ in terms of capturing the topological consistency of cross-network node pairs. As illustrated in Fig. 4, consider two networks G_s and G_t with $n_s = 6$ and $n_t = 4$, where there is a single matched pair (B,b) such that $\mathcal{T}^{(1)}_A = \{b\}$, $X^{(1)}_A = \{b, C, D, E, F\}$, $Y^{(1)}_a = \{b, c, d\}$, and $X^{(1)}_A \cap Y^{(1)}_a = \{b\}$. Suppose that $\alpha = \frac{1}{2}$ and $\beta = 1$. Then, we have $\mathbf{S}^{(1)}_{Tve}(A,a) = \frac{1}{5}$ and $\mathbf{S}^{(1)}_{Jac}(A,a) = \frac{1}{7}$. At the next gradual alignment step, we assume that (C,c) is newly aligned, which leads to $X^{(2)}_A = \{b, c, D, E, F\}, Y^{(2)}_a = \{b, c, d\}$, and $X^{(2)}_A \cap Y^{(2)}_a = \{b, c\}$. Then, it follows that $\mathbf{S}^{(2)}_{Tve}(A,a) = \frac{4}{9}$ and $\mathbf{S}^{(2)}_{Jac}(A,a) = \frac{2}{7}$. Since the growth rate of the Tversky similarity and the Jaccard index with respect to the number of ACNs is given by $\frac{11}{45}$ and $\frac{1}{7}$, respectively, using the Tversky similarity is beneficial in terms of better capturing the structural consistency across different networks, which finally leads to performance improvement.

After calculating our dual-perception similarity matrix $\mathbf{S}^{(i)}$ in (2) based on two similarity matrices $\mathbf{S}_{emb}$ and $\mathbf{S}^{(i)}_{Tve}$, we gradually match N node pairs for each iteration until all M node pairs are found. Let us explain how to choose N node pairs per iteration as follows. At the first iteration, we find the most confident pair (i.e., an element with the highest value) out of $n_s n_t$ elements in $\mathbf{S}^{(1)}$ and then find the second most confident pair among $(n_s - 1)(n_t - 1)$ elements after deleting all elements related to the most confident pair. We repeatedly perform the above steps until N node pairs are matched for the first iteration. Updating the node mapping $\pi^{(i)}$, we recalculate $\mathbf{S}^{(i+1)}_{Tve}$ to discover N node pairs at the next iteration. This process is repeated at each iteration. We refer to lines 18–22 in Algorithm 1 for calculation of $\mathbf{S}^{(i+1)}$ upon the updated one-to-one node mapping $\pi^{(i+1)}$.

4.2 Complexity Analysis

Theorem 4.3. *The computational complexity of the proposed Grad-Align method is given by* $\mathcal{O}(\max\{|\mathcal{E}_s|, |\mathcal{E}_t|\})$.

Proof. For a given node pair (u, v), we focus on analyzing the computational complexity of the Tversky similarity calculation. Since we take into account one-hop neighbors of each node in handling each term in (9), the complexity of calculating $|X_u^{(i)} - Y_v^{(i)}|$, $|Y_v^{(i)} - X_u^{(i)}|$, and $|X_u^{(i)} \cap Y_v^{(i)}|$ is given by $\mathcal{O}(|X_u^{(i)}|)$, $\mathcal{O}(|Y_v^{(i)}|)$, and $\mathcal{O}(|X_u^{(i)}||Y_v^{(i)}|)$, respectively, for the worst case, which are bounded by the maximum node degree in G_s and G_t and thus are regarded as a constant. As in the calculation of $\mathbf{S}_{emb}$, the Tversky similarity matrix $\mathbf{S}_{Tve}^{(i)}$ is computable in constant time due to the independent element-wise calculation via parallel processing. From the fact that the number of iterations for gradual alignment is finite and the computation of $\mathbf{S}_{emb}$ is a bottleneck, the total complexity of our Grad-Align method is finally bounded by $\mathcal{O}(\max\{|\mathcal{E}_s|, |\mathcal{E}_t|\})$. This completes the proof of this theorem. □

From Theorem 4.3, one can see that the computational complexity of Grad-Align scales linearly with the maximum number of edges over two networks. This also implies that the computational complexity of Grad-Align scales no larger than that of the embedding similarity calculation using the underlying GNN model.

4.3 Grad-Align-EA Method

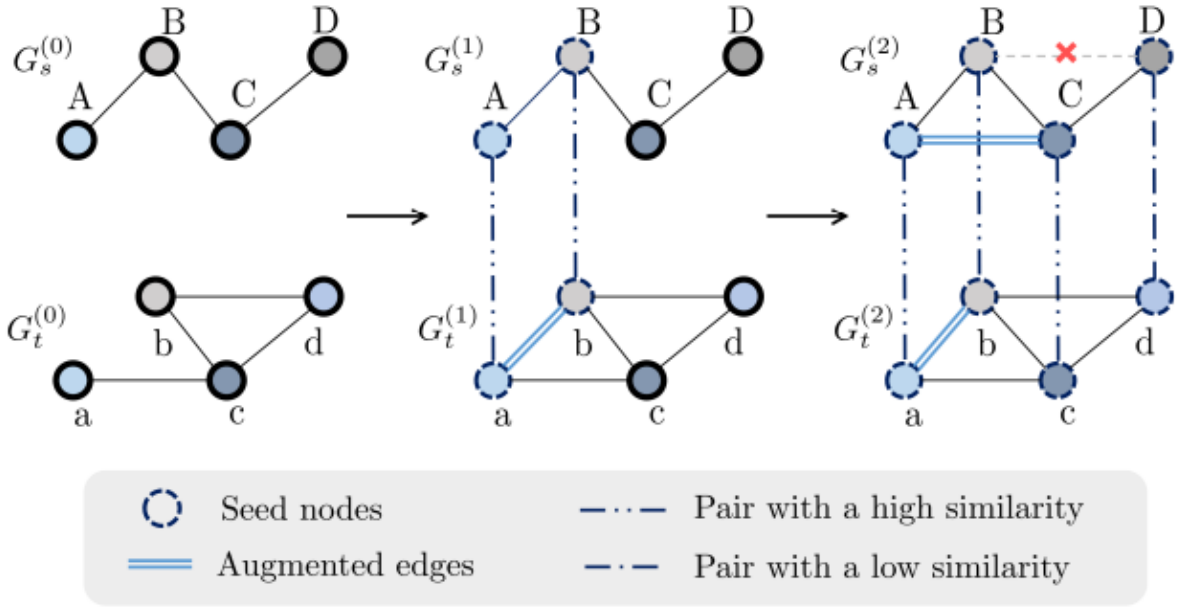

Fig. 5: An example illustrating edge augmentation over iterations. Here, the edge between nodes (a, b) in G_t is augmented at the first step, and the edge between nodes (A, C) is augmented at the second step.

In this subsection, we introduce Grad-Align-EA, namely Grad-Align with edge augmentation, to further improve the performance of Grad-Align by incorporating an edge augmentation module into the Grad-Align method, which is inspired by CENALP [19] that jointly performs NA and link prediction. However, different from the link prediction

More specifically, we augment edges only when the dual-perception similarity $\mathbf{S}^{(i)}(u, v)$ of node pair (u, v) is greater than a certain threshold $\tau > 0$ (refer to lines 3 and 9 in Algorithm 2), where τ will be set appropriately in Section 5.5.5. Fig. 5 illustrate an example of edge augmentation over iterations in our Grad-Align-EA. In the first step, the potentially confident edge between nodes (a, b) in G_t is augmented. In the second step, the edge between nodes (A, C) is augmented but another potentially confident edge between nodes (B, D) is not augmented because $\mathbf{S}^{(i)}(D, d)$ is not sufficiently high.

Since the structure of each network evolves over iterations via the edge augmentation module, we need to newly update the hidden representations of each network by re-training model parameters θ of GNN and then recalculate the multi-layer embedding similarity matrix. In this context, the dual-perception similarity matrix in (2) is rewritten as

$$\mathbf{S}^{(i)} = \mathbf{S}^{(i)}_{emb} \odot \mathbf{S}^{(i)}_{Tve}, \tag{14}$$

in CENALP that requires the high computational cost, our edge augmentation only creates edges that are *highly confident* without any training and test phases. The edge augmentation makes both networks G_s and G_t evolve in such a way that their structural consistencies across the networks are getting high over iterations in the gradual alignment process. To express the evolution of each network, we rewrite source and target networks as $G_s^{(i)} = (\mathcal{V}_s, \mathcal{E}_s^{(i)}, \mathcal{X}_s)$, $G_t^{(i)} = (\mathcal{V}_t, \mathcal{E}_t^{(i)}, \mathcal{X}_t)$, respectively, where $\mathcal{E}_s^{(i)}$ and $\mathcal{E}_t^{(i)}$ are the sets of edges in $G_s^{(i)}$ and $G_t^{(i)}$, respectively. The overall procedure of our edge augmentation is summarized in Algorithm 2.[8] Let $\tilde{\mathcal{E}}_s$ ($\tilde{\mathcal{E}}_t$) denote the set of *potentially confident edges* that are not in the set $\mathcal{E}_s^{(i)}$ ($\mathcal{E}_t^{(i)}$) while their counterpart edges upon the node mapping $\pi^{(i)}$ are in $\mathcal{E}_t^{(i)}$ ($\mathcal{E}_s^{(i)}$). If the aligned node pair does not match the ground truth, then the augmented edge will be treated as noise in the next alignment step. To remedy this problem, our edge augmentation module creates edges among potentially confident ones only when they are strongly confident.

Pesudo code

Algorithm 2 Edge augmentation

Input: $G_s^{(i)} = (\mathcal{V}_s, \mathcal{E}_s^{(i)}, \mathcal{X}_s)$, $G_t^{(i)} = (\mathcal{V}_t, \mathcal{E}_t^{(i)}, \mathcal{X}_t)$, $\pi^{(i)}$, $\mathbf{S}^{(i)}$, $\tilde{\mathcal{E}}_s$, $\tilde{\mathcal{E}}_t$, τ
Output: $G_s^{(i+1)}, G_t^{(i+1)}$

/* Edge augmentation for $G_s^{(i)}$ */
for $(u_1, u_2) \in \tilde{\mathcal{E}}_s$ **do**
  if $\mathbf{S}^{(i)}(u, \pi^{(i)}(u)) > \tau$ and $\mathbf{S}^{(i)}(v, \pi^{(i)}(v)) > \tau$ **then**
    $\mathcal{E}_s^{(i+1)} = \mathcal{E}_s^{(i)} \cup \{(u, v)\}$
  end if
end for
/* Edge augmentation for $G_t^{(i)}$ */
for $(v_1, v_2) \in \tilde{\mathcal{E}}_t$ **do**
  if $\mathbf{S}^{(i)}(\pi^{-1(i)}(u'), u') > \tau$ and $\mathbf{S}^{(i)}(\pi^{-1(i)}(v'), v') > \tau$ **then**
    $\mathcal{E}_t^{(i+1)} = \mathcal{E}_t^{(i)} \cup \{(u', v')\}$
  end if
end for
return $G_s^{(i+1)} = (\mathcal{V}_s, \mathcal{E}_s^{(i+1)}, \mathcal{X}_s)$, $G_t^{(i+1)} = (\mathcal{V}_t, \mathcal{E}_t^{(i+1)}, \mathcal{X}_t)$

Datasets		# of nodes	# of edges	# of attributes	# of ground truth node pairs
Facebook	G_s	1,043	4,734	0	1,043
Twitter	G_t	1,043	4,860	0	
Douban Online	G_s	3,906	8,164	538	1,118
Douban Offline	G_t	1,118	1,511	538	
Allmovie	G_s	6,011	124,709	14	5,176
IMDb	G_t	5,713	119,073	14	
Facebook	G_s	1,256	4,260	0	1,043
(Its noisy version)	G_t	1,256	4,256	0	
Econ	G_s	1,258	6,857	20	1,258
(Its noisy version)	G_t	1,258	6,860	20	
DBLP	G_s	2,151	5,676	20	2,158
(Its noisy version)	G_t	2,151	5,672	20	
Foursquare	G_s	17,355	132,208	0	17,355
(Its noisy version)	G_t	17,355	131,018	0	

TABLE 2: Statistics of the seven datasets used in our experiments.

where $\mathbf{S}_{emb}^{(i)}$ represents the multi-layer embedding similarity matrix at the i-th iteration when Grad-Align-EA is used.