Appendix of S3GA: Towards Scalable Self-Supervised Learning for Large-Scale Graph Alignment

A Theoretical Analysis

A.1 Proof of Proposition 2.1

Proposition A.1. When solving the large-scale GA problem under a divide-and-conquer strategy, the predicted alignment accuracy is always less than or equal to the alignment-aware partition accuracy:

$$alignment\_acc \leq AAP\_acc. \tag{22}$$

Proof. Let the $\mathcal{S}^* = \{(v_i, u_j) \mid (v_i, u_j) \in \mathcal{V}_s \times \mathcal{V}_t, v_i \equiv u_j\}$ is the ground-truth aligned node pair set. According to the definition of *alignment-aware partition accuracy*, we have

$$\text{AAP\_acc} = \frac{1}{|\mathcal{S}^*|} \sum_{k=0}^{N_c-1} |\mathcal{S}_k \cap \mathcal{S}^*| \tag{23}$$

$$= \frac{1}{|\mathcal{S}^*|} \sum_{k=0}^{N_c-1} |\mathcal{S}_k \cap (\mathcal{S}^*_{\text{intra}} \cup \mathcal{S}^*_{\text{inter}})| \tag{24}$$

$$= \frac{1}{|\mathcal{S}^*|} \sum_{k=0}^{N_c-1} |(\mathcal{S}_k \cap \mathcal{S}^*_{\text{intra}}) \cup (\mathcal{S}_k \cap \mathcal{S}^*_{\text{inter}})| \tag{25}$$

$$= \frac{1}{|\mathcal{S}^*|} \sum_{k=0}^{N_c-1} |\mathcal{S}_k \cap \mathcal{S}^*_{\text{intra}}| \tag{26}$$

$$= \frac{1}{|\mathcal{S}^*|} \sum_{k=0}^{N_c-1} |\left\{(v_i, u_j) \mid v_i \in \mathcal{V}_{s_k}, u_j \in \mathcal{V}_{t_k}, v_i \equiv u_j\right\}| \tag{27}$$

$$\geq \frac{1}{|\mathcal{S}^*|} \sum_{k=0}^{N_c-1} |\left\{(v_i, u_j) \mid (v_i, u_j) \in \mathcal{V}_{s_k} \times \mathcal{V}_{t_k}, v_i \equiv u_j, v_i \equiv^{\theta} u_j\right\}| \tag{28}$$

$$= \text{alignment\_acc}. \tag{29}$$

Equality holds if and only if the network prediction accuracy is 100%. □

A.2 Proof of Theorem 3.1

Theorem A.1 (Convergence Guarantee of AAC). Let $\mathcal{G}_s$ and $\mathcal{G}_t$ be two input graphs, and let $C^{s(n)}$ and $C^{t(n)}$ be the cluster assignments generated by AAC at the n-th iteration. The Alignment-Aware Clustering (AAC) process is guaranteed to converge to a stable partition after a finite number of iterations. Specifically, there exist a step N such that

$$\forall n \geq N, \quad C^{s(n)} = C^{s(n+1)} \quad \textit{and} \quad C^{t(n)} = C^{t(n+1)}.$$

Proof. Given the loss function defined as:

$$J = -\sum_{k=1}^{N_c} \sum_{(v_m, v_n) \in C_k} \cos(\mathbf{X}^s[v_m, :], \mu_k) + \cos(\mathbf{X}^t[v_n, :], \mu_k), \tag{30}$$

where N_c represents the number of cluster, and (v_m, v_n) denotes pairs of nodes assigned to cluster C_k. Here, $\mathbf{X}^S[v_m, :]$ and $\mathbf{X}^T[v_n, :]$ are the embeddings of nodes v_m and v_n from the source and target graphs, respectively, while μ_i is the centroid of cluster C_i.

To prove the convergence of the AAC algorithm, a sufficient condition is that the loss function defined above is both monotonic and bounded.

First, consider **monotonicity**. Without loss of generality, consider the loss function for a single cluster k. Based on the *Inliers* and *Outliers*, the loss function can be summed as:

$$J_k = J_k^{\mathbb{I}} + J_k^{\mathbb{O}}, \tag{31}$$

$$\text{with} \quad J_k^{\mathbb{I}} = -\sum_{(v_m, v_n) \in \mathbb{I}_k} \cos(\mathbf{X}^s[v_m, :], \mu_k) + \cos(\mathbf{X}^t[v_n, :], \mu_k),$$

$$\text{and} \quad J_k^{\mathbb{O}} = -\sum_{(v_m, v_n) \in \mathbb{O}_k} \cos(\mathbf{X}^s[v_m, :], \mu_k) + \cos(\mathbf{X}^t[v_n, :], \mu_k) \tag{32}$$

Here, $\mathbb{I}_k$ and $\mathbb{O}_k$ represent the sets of *Inliers* and *Outliers*, respectively.

The AAC algorithm can be divided into two main steps:

Step 1: Centroid Update using *Inliers*. Then we differentiate the loss function J_k with respect to μ_k:

$$\frac{\partial J_k^{\mathbb{I}}}{\partial \mu_k} = -\frac{\partial}{\partial \mu_k} \sum_{(v_m, v_n) \in \mathbb{I}_k} \left(\frac{\mathbf{X}^s[v_m, :] \cdot \mu_k}{\|\mu_k\|} + \frac{\mathbf{X}^t[v_n, :] \cdot \mu_k}{\|\mu_k\|} \right) \tag{33}$$

$$= -\sum_{(v_m, v_n) \in \mathbb{I}_k} \left[\frac{\mathbf{X}^s[v_m, :]}{\|\mu_k\|} - \frac{(\mathbf{X}^s[v_m, :] \cdot \mu_k)\, \mu_k}{\|\mu_k\|^3} \right] - \sum_{(v_m, v_n) \in \mathbb{I}_k} \left[\frac{\mathbf{X}^t[v_n, :]}{\|\mu_k\|} - \frac{(\mathbf{X}^t[v_n, :] \cdot \mu_k)\, \mu_k}{\|\mu_k\|^3} \right]. \tag{34}$$

The first equality holds because we assume the node embedding vectors have already been normalized. To find the optimal centroid μ^k, we set the derivative to zero:

$$\sum_{(v_m, v_n) \in \mathbb{I}_k} \left[\frac{\mathbf{X}^s[v_m, :]}{\|\mu_k\|} + \frac{\mathbf{X}^t[v_n, :]}{\|\mu_k\|} \right] = \sum_{(v_m, v_n) \in \mathbb{I}_k} \left[\frac{(\mathbf{X}^s[v_m, :] \cdot \mu_k)\, \mu_k}{\|\mu_k\|^3} + \frac{(\mathbf{X}^t[v_n, :] \cdot \mu_k)\, \mu_k}{\|\mu_k\|^3} \right]. \tag{35}$$

To simplify, we multiply both sides by $\|\mu_k\|$:

$$\sum_{(v_m, v_n) \in \mathbb{I}_k} \left[\mathbf{X}^s[v_m, :] + \mathbf{X}^t[v_n, :] \right] = \frac{\mu_k}{\|\mu_k\|} \sum_{(v_m, v_n) \in \mathbb{I}_k} \left[\frac{(\mathbf{X}^s[v_m, :] \cdot \mu_k)}{\|\mu_k\|} + \frac{(\mathbf{X}^t[v_n, :] \cdot \mu_k)}{\|\mu_k\|} \right] = \frac{\mu_k}{\|\mu_k\|} \sum_{(v_m, v_n) \in \mathbb{I}_k} \left[\cos(\mathbf{X}^s[v_m, :], \mu_k) + \cos(\mathbf{X}^t[v_n, :], \mu_k) \right] \tag{36}$$

Then, we can get that

$$\frac{\mu_k}{\|\mu_k\|} = \frac{\sum_{(v_m, v_n) \in \mathbb{I}_k} \left[\mathbf{X}^s[v_m, :] + \mathbf{X}^t[v_n, :] \right]}{\sum_{(v_m, v_n) \in \mathbb{I}_k} \left[\cos(\mathbf{X}^s[v_m, :], \mu_k) + \cos(\mathbf{X}^t[v_n, :], \mu_k) \right]} \tag{37}$$

In AAC, *Inliers* are considered to be nodes that are well-aligned with the centroid, meaning the cosine similarities between their

embeddings and the centroid are close to 1. Therefore, the denominator can be approximated as $2|\mathcal{I}_k|$. By omitting the normalization, the expression simplifies to match eq. (10) that:

$$\mu_k^{(n+1)} = \frac{1}{2|\mathcal{I}_k|} \sum_{(v_m,v_n)\in\mathcal{I}_k} \left(X_k^s[v_m,:] + X_k^t[v_n,:]\right). \tag{38}$$

The updated μ_k corresponds to the minimum point of the function, ensuring that this update step does not increase the loss function, i.e.,

$$J^{\mathbb{I}(n+1)} \leq J^{\mathbb{I}(n)}. \tag{39}$$

Remark 1. The approximation used in eqs. (37) and (38),

$$\sum_{(v_m,v_n)\in\mathbb{I}_k} \left[\cos\left(\mathbf{X}^s\left[v_m,:\right],\mu_k\right) + \cos\left(\mathbf{X}^t\left[v_n,:\right],\mu_k\right)\right] \approx 2|\mathcal{I}_k|,$$

simplifies the denominator for tractable computation. Although this approximation may not strictly hold in practice, it leads to an update direction $\Delta\boldsymbol{\mu}_k = \boldsymbol{\mu}_k^{(n+1)} - \boldsymbol{\mu}_k^{(n)}$ that still satisfies the descent condition $\mathbf{J}(\boldsymbol{\mu}_k)^\top \Delta\boldsymbol{\mu}_k < 0$.

Step 2: *Outliers* are reassigned to the clusters that maximize their cosine similarity with the updated centroids, which further reduces the loss function:

$$J^{\mathbb{O}(n+1)} \leq J^{\mathbb{O}(n)}. \tag{40}$$

By iteratively updating the centroids and reassigning nodes, the loss function J either decreases or remains constant after each step.

Secondly, let's consider the **boundedness** of J. Clearly, J is bounded below by $-\sum_{k=1}^{N_c} |C_k^s| + |C_k^t|$, where $|C_k^s|$ and $|C_k^t|$ denote the number of source and target nodes in cluster k, respectively.

Thus, this iterative process is guaranteed to converge after a finite number of iterations. Consequently, there exists a step N such that for all $n \geq N$,

$$C^{s(n)} = C^{s(n+1)} \quad \text{and} \quad C^{t(n)} = C^{t(n+1)}.$$

□

Proof of Remark 1. We aim to verify that the practical update

$$\Delta\boldsymbol{\mu}_k = \boldsymbol{\mu}_k^{(n+1)} - \boldsymbol{\mu}_k^{(n)}$$

satisfies the descent condition

$$\mathbf{J}(\boldsymbol{\mu}_k)^\top \Delta\boldsymbol{\mu}_k < 0.$$

We assume $\cos(\mathbf{X}, \mu_k) > 0$, i.e., the inlier nodes are well-aligned with the cluster center, so that both

$$a = \mathbf{X}^s[v_m,:]^\top \mu_k > 0, \quad b = \mathbf{X}^t[v_n,:]^\top \mu_k > 0.$$

1. Direction-preserving gradient. We define a scaled gradient preserving the descent direction:

$$\widetilde{\nabla}_{\mu_k} J = \|\mu_k\| \nabla_{\mu_k} J = -\sum_{(v_m,v_n)\in\mathbb{I}_k} \left(\mathbf{X}^s[v_m,:] + \mathbf{X}^t[v_n,:] - \frac{(a+b)\mu_k}{\|\mu_k\|^2}\right).$$

2. Inner product descent condition. Then,

$$\widetilde{\nabla}_{\mu_k}^\top \Delta\mu_k = -\frac{1}{2|\mathbb{I}_k|} \sum_{(v_m,v_n)\in\mathbb{I}_k} \left\|\mathbf{X}^s[v_m,:] + \mathbf{X}^t[v_n,:]\right\|^2 - \frac{(a+b)^2}{\|\mu_k\|^2}.$$

Since all terms are positive, the sum is negative:

$$\widetilde{\nabla}_{\mu_k}^\top \Delta\mu_k < 0.$$

3. Stationary condition. We further observe:

$$\begin{aligned} -\widetilde{\nabla}_{\mu_k}^\top \mu_k^{(n+1)} &= \sum_{(v_m,v_n)\in\mathbb{I}_k} \left\langle \mathbf{X}^s[v_m,:] + \mathbf{X}^t[v_n,:], \mu_k \right\rangle - \frac{(a+b)}{\|\mu_k\|^2}\|\mu_k\|^2 \\ &= \sum (a+b-(a+b)) = 0. \end{aligned}$$

4. By Cauchy-Schwarz inequality. Letting $\mathbf{u} = \mathbf{X}^s[v_m,:] + \mathbf{X}^t[v_n,:]$, $\mathbf{v} = \mu_k$, and using

$$(\mathbf{u}^\top \mathbf{v})^2 \leq \|\mathbf{u}\|^2 \|\mathbf{v}\|^2,$$

we derive:

$$\frac{(a+b)^2}{\|\mu_k\|^2} \leq \left\|\mathbf{X}^s[v_m,:] + \mathbf{X}^t[v_n,:]\right\|^2,$$

with equality only when all vectors are exactly aligned with μ_k, which is statistically improbable. Thus, the inequality holds strictly in most cases.

Therefore, both $\nabla_{\mu_k}^\top \Delta\mu_k < 0$ and $\widetilde{\nabla}_{\mu_k}^\top \Delta\mu_k < 0$ hold, and the descent condition is satisfied.

A.3 Theoretical Analysis of the Effectiveness of the Loss Function (Eq. 18)

Proposition A.2. Any normalized vector $\boldsymbol{z}$ gives a quadratic score that obeys the following optimality bound [19]:

$$\frac{\boldsymbol{z}^\top \boldsymbol{K}(\boldsymbol{w})\boldsymbol{z}}{\boldsymbol{z_{opt}}(\boldsymbol{w})^\top \boldsymbol{K}(\boldsymbol{w})\boldsymbol{z_{opt}}(\boldsymbol{w})} \geq 2\left(\boldsymbol{z}^\top \mathbf{V}(\mathbf{w})\right)^2 - 1, \tag{41}$$

where $\boldsymbol{z_{opt}}(\boldsymbol{w})$ is the optimal solution to the QAP for a given parameter $\boldsymbol{w}$, and $\boldsymbol{V}(\boldsymbol{w})$ is the eigenvector of the $\boldsymbol{K}(\boldsymbol{w})$.

Accordingly, it immediately follows that maximizing $\boldsymbol{z}^\top \boldsymbol{V}(\boldsymbol{w})$ would maximize the lower bound in Eq. (41) to approach to 1, pushing $\boldsymbol{z}$ to approximate $\boldsymbol{z_{opt}}(\boldsymbol{w})$. We perform the Sinkhorn or RCF algorithm in Eq. (17) to tackle the linear assignment problem. According to [19], it is reasonable to use the vectorized probability score $\mathrm{vec}(softmax(\mathbf{M}))$ to replace the eigenvector $\boldsymbol{V}(\boldsymbol{w})$ in Eq. (41). Then, the computed $\widehat{\mathbf{P}}$ in Eq. (17) can be further used by maximizing the similarity score $\mathrm{tr}(\widehat{\mathbf{P}}^\top \mathrm{softmax}(\mathbf{M}))$ to train the network parameters. In practice, we use the cross-entropy in Eq. (18).

B Pseudo Code for Algorithms

C More Experiments

C.1 Datasets Details

We conduct experiments on datasets with different sizes from three cross-lingual GA benchmarks, namely IDS100K [34], DBP15K [33] and DBP1M [10]. The dataset statistics are shown in Tab.(7).

- IDS100K includes two cross-lingual knowledge graph datasets that pair English with French and English with German, generated using the IDS algorithm [34]. This benchmark incorporates encoded URIs for entities instead of entity names to mitigate potential biases associated with the names. Consistent with prior research [9, 10], we employ version 1.0, which adheres to the data distribution characteristics of actual knowledge graphs and retains nodes with low degrees.
- DBP1M stands as the largest cross-lingual GA benchmark, featuring two extensive datasets derived from DBpedia [2] that pair English with French and English with German. This

Algorithm 1 Alignment-Aware Clustering (AAC)

Input: Node feature matrices $\mathbf{X}_s$, $\mathbf{X}_t$; cluster sets C_s, C_t consisting of N_c clusters, where each k-th element C_k^s, C_k^t contains the source and target nodes in cluster k; maximum iterations max_iter.

Output: Pseudo-aligned node pair set $\mathbb{I}$.

$iter \leftarrow 0$
while not converged and $iter < max\_iter$ **do**
  Initialize a set $\mathbb{U}$ to store unassigned nodes.
  for $k = 0$ **to** $N_c - 1$ **do**
    Add noisy nodes to C_k^s and C_k^t, forming $\hat{C}_k^s$ and $\hat{C}_k^t$. (See Eq. 8).
    Compute the aligned node pair set $\mathbb{P}_k$ between $\hat{C}_k^s$ and $\hat{C}_k^t$ based on their features $\mathbf{X}_k^s$ and $\mathbf{X}_k^t$. (See Eq. 9).
    Identify the node pairs *inliers* $\mathbb{I}_k$ and *outliers* $\mathbb{O}_k$ from $\mathbb{P}_k$ according to their definitions. (See Def. 2.2 and Def. 2.3).
    Update the cluster centroid μ_k with the mean embeddings of nodes in $\mathbb{I}_k$. (See Eq. 10)
    Retain only the nodes in *inliers* within cluster k and remove the others.
    Add non-noisy nodes from $\mathbb{O}_k$ to $\mathbb{U}$.
  end for
  Assign each node in $\mathbb{U}$ to the cluster with the closest updated centroid and combine them with the original *inliers* to form the new cluster sets C_s and C_t. (see Eq. 11 and Eq. 12).
  $iter \leftarrow iter + 1$.
end while

Algorithm 2 RCF for Pseudo Label Generation

Input: Affinity matrix $\widehat{\mathbf{M}} \in \mathbb{R}^{n_s \times n_t}$.

Output: Pseduo label matrix $\widehat{\mathbf{P}}$.

Initialize $\mathbf{r} \in \{1\}^{n_s}$, $\mathbf{c} \in \{1\}^{n_t}$, and $\widehat{\mathbf{P}} \in \mathbb{R}^{n_s \times n_t}$ as all-zero
Compute $\mathbf{S} = \text{Sinkhorn}(\widehat{\mathbf{M}})$
while $\mathbf{S}$ is non-empty **do**
  Initialize candidate set $\mathbb{Z}$
  Collect row-wise and column-wise argmax indices into $\mathbb{Z}$
  for each $[i, j]$ in $\mathbb{Z}$ that appears twice **do**
    Set $\widehat{\mathbf{P}}[i, j] = 1$; mask out row i and column j
    Update $\mathbf{r}[i] = \mathbf{c}[j] = 0$
  end for
  while $\mathbb{Z} \neq \emptyset$ **do**
    For each $[i, j] \in \mathbb{Z}$, compute row confidence: Δ = 1st max − 2nd max
    Select $[i, j]$ with highest Δ, set $\widehat{\mathbf{P}}[i, j] = 1$, remove conflicting indices
    Update $\mathbf{r}[i] = \mathbf{c}[j] = 0$
  end while
  Prune $\mathbf{S}$ by row/col masks: $\mathbf{S} = \mathbf{S}[\mathbf{r}, \mathbf{c}]$
end while

benchmark includes isolated nodes and exhibits biases related to name information. In alignment with the Clusterea approach, we exclude non-matchable nodes from consideration.

- DBP15K is a smaller-scale cross-lingual GA benchmark derived from DBpedia, pairing English with French, German, and Chinese. This dataset is commonly used to evaluate graph alignment methods due to its manageable size and well-defined structure, offering a balanced mix of entities and relationships. Previous works always performed matching on nodes with corresponding relationships in the 15K dataset, excluding outliers. In contrast, we test the alignment in the presence of outliers, involving all nodes in the matching process.

C.2 Baseline methods

- GCNAlign [43]: As the pioneering GNN-based entity alignment model, GCNAlign leverages GCN to compute node embeddings, establishing pseudo-node correspondences through cross-graph embedding similarity.
- Dual-AMN [25]: An entity alignment model that utilizes a Proxy Matching Attention Layer to effectively manage cross-graph relations.
- LargeEA [10]: Recognized as the first GA framework designed for scalability, LargeEA operates by training multiple EA models on mini-batches generated via a rule-based strategy, accommodating large-scale graph data.
- ClusterEA* [9]: a SOTA method for large-scale supervised learning in GA. It incorporates stochastic training, CluserSampler, and SparseFusion to enhance GA performance. In our modification, ClusterEA*, during the second cluster & classification phase, we removed the use of seed nodes for directly aligning the two graphs. Instead, we first train on one graph and then make predictions on the other graph. Additionally, to enhance generalization, we incorporated the Optimal Transport loss, a domain adaptation technique, to better align the feature distributions between the source and target graphs.
- LightEA* [27]: a non-neural framework that reimplements the label propagation algorithm for efficient entity alignment on knowledge graphs. However, its third-order representation scales as $O(N^3)$, making it unsuitable for graphs with high-dimensional feature representations.

D Further Discussions

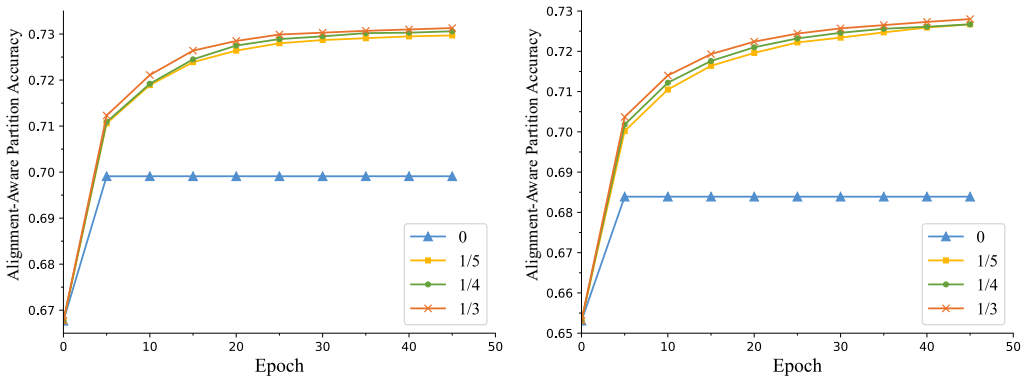
Why do we choose the RCF algorithm as the graph-matching solver? The Rival Confidence First (RCF) algorithm is proposed to solve the Eq. (1) by adapting the Sinkhorn algorithm [4] to generate the results that meet the one-to-one constraint. The Sinkhorn algorithm solves the Eq. (1) by adding an entropy regularization:

$$\begin{aligned} \mathbf{S}^* &= \underset{\mathbf{S}}{\text{argmax}}\ \text{tr}\left(\mathbf{S}^\top \mathbf{M}\right) + \tau h(\mathbf{S}) \\ \text{s.t.} \quad & \mathbf{S} \in [0,1]^{n_1 \times n_2}, \mathbf{S}\mathbf{1}_{n_2} = \mathbf{1}, \mathbf{S}^\top \mathbf{1}_{n_1} \leq \mathbf{1}, \end{aligned} \tag{42}$$

where $h(\mathbf{S}) = \sum_{i,j} \mathbf{S}_{ij} \log \mathbf{S}_{ij}$ is the entropic regularizer. In this approach, the binary constraint on $\mathbf{P}$ in Eq. (1) is relaxed to allow continuous values within $[0, 1]$. Consequently, the optimal solution $\mathbf{S}^*$ obtained is usually infeasible for the original problem. The gap between two optimal solution, measured by $||\mathbf{P}^* - \mathbf{S}^*||_F$, has an

Table 7: Dataset statistics .

Datasets		#nodes	#edges
DBP1M	EN-FR	EN: 1,877,793; FR: 1,365,118	EN: 7,031,172; FR: 2,997,457
	EN-DE	EN:1,625,999; DE:1,112,970	EN:6,213,639; DE:1,994,876
IDS100K	EN-FR	EN: 100,000; FR: 100,000	EN: 309,607; FR: 258,285
	EN-DE	EN: 100,000; DE: 100,000	EN: 335,359; DE: 336,240
DBP15K	ZH-EN	ZH: 19,388; EN: 19,572	ZH: 70,414; EN: 95,142
	JA-EN	JA: 19,814; EN: 19,780	JA: 77,214; EN: 93,484
	FR-EN	FR: 19,661; EN: 19,993	FR: 39,581; EN: 44,804



(a) DBP1M$_{\text{EN-FR}}$ (b) DBP1M$_{\text{EN-DE}}$

Figure 5: The alignment-aware partition accuracy with varying noisy node proportions relative to cluster size in the DBP1M dataset. A value of 0 indicates no noise added. After one epoch, the partition accuracy remains unchanged.

upper bound give by

$$\frac{\tau\left(\ln n_1+\ln n_2\right)}{\sum_{i=1}^{n_1}\left(\mathbf{M}_{i,\varnothing_i}-\mathbf{M}_{i,\varnothing_{i+1}}\right)}, \tag{43}$$

where $\mathbf{M}_{i,\varnothing_i}$ and $\mathbf{M}_{i,\varnothing_{i+1}}$ denotes the largest and the second largest element of row i in $\mathbf{M}$, respectively. Utilizing this proposition, the RCF algorithm aims to minimize the upper bound of the discrepancy $||\mathbf{P}^*-\mathbf{S}^*||_F$, thereby reducing the gap between the relaxed and the original optimal solutions. This approach generates a solution that satisfies the one-to-one constraint and closely approximates the optimal solution $\mathbf{P}^*$. Experimentally, we compare the GA performance of the RCF with the Sinkhorn algorithm on the DBP1M dataset. The results show that the RCF algorithm generates more accurate alignment results.

How does alignment accuracy perform during the self-supervised learning process? Lastly, we assess the alignment accuracy during the self-supervised training process on two datasets. Fig. 6(a) and Fig. (b) show that the GCN effectively learns node embeddings, leading to more accurate alignments. This underscores the crucial role of our TAR algorithm in preserving topology information for nodes, which is essential for effective GCN learning.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

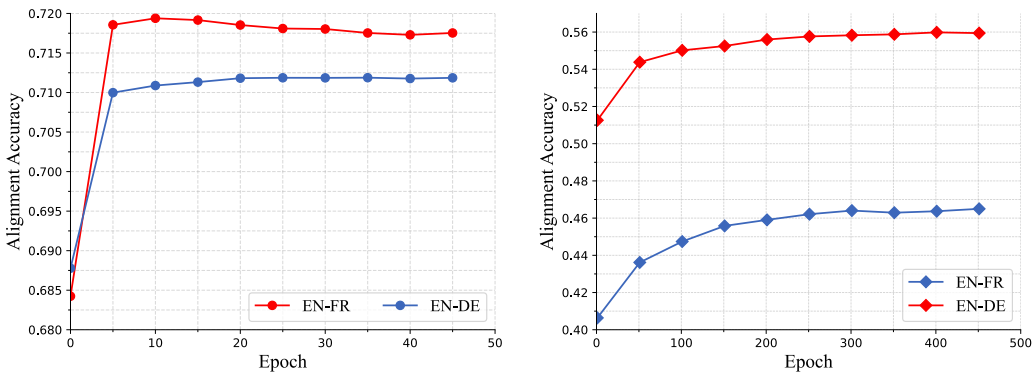


Figure 6: Fig. (a) and Fig. (b) illustrate the variation in alignment accuracy during the self-supervised learning phases of the small-scale GA, process for the DBP1M and IDS100K datasets, respectively.

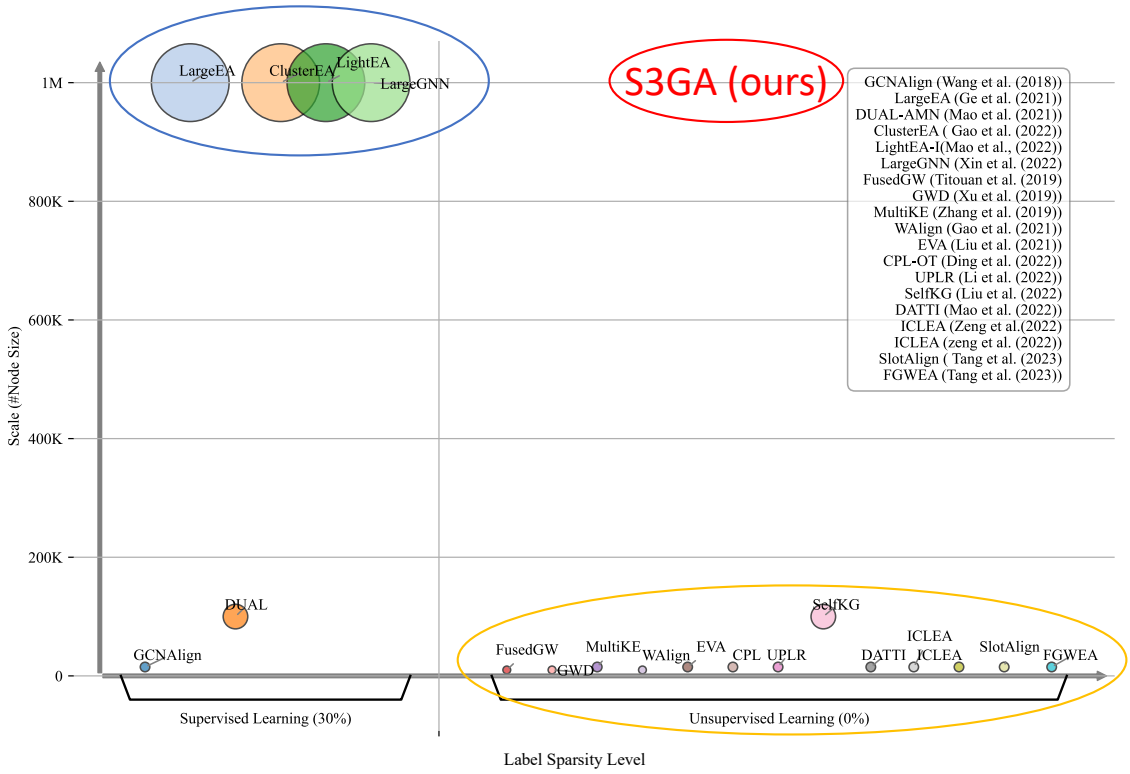


Figure 7: Comparison of various graph alignment methods based on scale (number of nodes) and label dependency level. Methods in the top-left area, such as LargeEA, ClusterEA, and LightEA, can handle large-scale graphs (over 1M nodes) but require supervised learning. In contrast, methods in the bottom-right area, such as SelfKG, operate under unsupervised learning and handle smaller-scale graphs. Currently, no unsupervised graph alignment (GA) method can handle large-scale graphs with millions of nodes.