Computers and Artificial Intelligence；Vol.2, No.3; 2025
E-ISSN: 3006 -3728 P-ISSN：3006-371X
DOI: https://doi.org/10.70267/ cai.25v2n3.158166
Published by: Zeus Press

SSGTN: A Graph Similarity Calculation Method Combining Subgraph and Global Features

Huang Xu[*]

Guizhou University of Finance and Economics, Guiyang, Guizhou, 550025, China

**Corresponding author: Huang Xu*.*

Abstract

Graph similarity computation is one of the fundamental challenges in graph data mining, with critical applications in chemical molecule analysis and social network analysis. Traditional methods, such as graph edit distance and maximum common subgraph, are NP-hard and computationally expensive. Although neural network-based approaches have improved efficiency and accuracy, existing methods still exhibit limitations: they often focus excessively on global features while neglecting local substructures, or vice versa, and fail to evaluate the relative importance of node-level, subgraph level, and global features. To address these issues, this paper proposes Similarity Subgraph Global Tensor Network(SSGTN), a novel graph similarity computation framework that integrates subgraph and global features. Specifically, SSGTN employs graph convolutional networks to extract subgraph features and graph attention mechanisms to capture global representations. A Subgraph Global Tensor Network is then designed to dynamically fuse pooled subgraph features with global features, thereby jointly optimizing local and global characterizations. Finally, all features are aggregated to generate the graph similarity score. Experiments on real-world datasets demonstrate that SSGTN outperforms baseline models across three evaluation metrics, confirming its efficiency and accuracy.

Keywords

graph neural network, graph similarity calculation, graph embedding, graph editing distance, maximum common subgraph

1. Introduction

As a specialized data structure, graphs possess unique advantages over other data structures for storing data in fields such as chemical molecules, social networks, medical pharmaceuticals, and computer security. A fundamental problem closely related to graphs is graph similarity computation. This task is not only a core operation in graph database analysis, graph classification, and other direct applications in the graph domain, but is also widely applied across various disciplines. For instance, in social network analysis, comparing the similarity of information graphs can reveal behavioral patterns of users (Koutra et al., 2013), greatly facilitating analytical tasks. In anomaly detection, constructing communication graphs and computing their similarity can significantly contribute to identifying network intrusions (Noble and Cook, 2003). In neuroscience, constructing brain networks and calculating their similarity aids in the study of brain disorders (Liu et al., 2019, Ma et al., 2021). Consequently, graph similarity computation stands as one of the most crucial research topics based on graphs, holding substantial theoretical value and practical significance.

Graph Edit Distance (GED) (Bunke., 1983) and Maximum Common Subgraph (MCS) (Bunke and Shearer,

1998) are two of the most widely adopted metrics for graph similarity computation. However, research has shown that computing the exact GED or MCS for two graphs is an NP-hard problem (Bunke and Shearer, 1998, Zeng et al., 2009), which imposes significant computational overhead in practice. To date, in state-of-the-art research, computing the exact GED for graphs with more than 16 nodes remains infeasible within a reasonable time frame (Blumenthal and Gamper, 2020). Confronted with the high value and considerable challenges of graph similarity computation, researchers have turned to deep learning methods to learn similarity measures directly from data. During training, these models learn parameters by minimizing the loss between predicted and ground-truth values, enabling rapid similarity computation. In recent years, Graph Neural Networks (GNNs), as a deep learning approach based on graph structures, have been extensively applied to various graph-related problems and have provided a solid theoretical foundation for graph similarity computation. Researchers have begun to frame graph similarity computation as a regression task, constructing neural network models to capture features between two graphs—either at the node level or the graph level—and subsequently compute their similarity(Bai et al., 2019, Bai et al., 2020, Zhang et al., 2021, Li et al., 2019, Berretti et al., 2001, Riba et al., 2018, Ktena et al., 2017, Wang et al., 2024, Zhuo and Tan, 2022, Berahmand et al., 2025).

While deep learning has greatly facilitated graph similarity computation, it also introduces new challenges. Most existing approaches primarily utilize GNNs to extract node features and global features, which are then combined for computation. Many methods represent the entire graph as a fixed-length vector, even though real-world graphs exhibit significant variations in size. These approaches still rely heavily on the similarity of graph-level embeddings, focusing more on global characteristics while overlooking local, substructural features of the graph data. Furthermore, the processing of node-level information is often relatively coarse. For simple graph data, such methods may suffice for reasonably accurate similarity computation. However, they often struggle to achieve satisfactory results in more complex scenarios. For example, when the input graphs possess highly complex structures, the model may fail to effectively capture global features. Alternatively, when two input graphs have similar sizes but their primary differences lie in local substructures, the model's overemphasis on global features can lead to significant errors in the computed similarity.

Therefore, current research faces the following challenges: (1) How to better focus on subgraph features without neglecting global features when dealing with data rich in subgraph characteristics and complex graph structures? (2) Given the size variations of graphs in the real world, how can we model them appropriately while preserving their original sizes? (3) Different node features, as well as the features of nodes, subgraphs, and the global structure, have varying degrees of importance. How can we effectively balance their importance while also considering computational complexity to achieve a significant improvement in accuracy?

To address the aforementioned challenges, this paper proposes a graph similarity computation method named SSGTN, which integrates a subgraph global tensor network. The innovations of this work are as follows:

(1) It is the first to propose a Subgraph Global Tensor Network that jointly extracts features from both subgraphs and the global graph structure, maximizing the utilization of subgraph features while also leveraging global characteristics.

(2) It integrates graph convolutional networks(GCN) and Neural Tensor Networks(NTN), preserving the original sizes of the input graphs during modeling.

(3) Subgraph structures of different sizes are processed separately, effectively balancing the importance of node features, subgraph features, and global features. Consequently, the model achieves improved accuracy across various datasets compared to existing models.

2. Related Work

2.1 Graph Convolutional Networks

GCN are a method for node embedding based on neighborhood aggregation (Ling et al., 2021). As shown in Figure 1, its core operation is the graph convolution. In this operation, the representation of each node in the next layer is computed as a weighted sum of the features from its neighboring nodes and itself in the previous layer. It follows that a single graph convolution layer can extract features from the node itself and its immediate neighborhood. After multiple layers, each node's representation incorporates information from all nodes in the graph. Since graph data inherently involves interdependencies among nodes, GCN has become one of the most fundamental algorithms in Graph Neural Networks.

2.2 Graph Attention Mechanism

Building upon GCN, graph attention networks(GAT) introduce an attention mechanism to achieve more effective neighbor aggregation and address the limitations of GCN's fixed structural dependencies and simplistic node contribution weighting (Kipf, 2016). By using attention mechanisms to compute updated node embedding vectors, GAT can partially or entirely decouple from the graph structure, establishing a more generalized paradigm compared to standard GCN.

2.3 Traditional Graph Similarity Computation

In traditional graph similarity computation, the primary task is to find a suitable distance metric. GED was first proposed in 1983 (Bunke., 1983), defining the distance between two graphs as the minimum cost sequence of edit operations required to transform one graph into the other. In 1998, MCS was proposed (Bunke and Shearer, 1998), which circumvented the need for explicitly finding edit paths. However, subsequent research demonstrated their equivalence (Veličković et al., 2018), implying that algorithms suitable for computing one are generally applicable to the other. Since computing these exact metrics is NP-hard (Zeng et al., 2009), it remains intractable to compute them directly for graphs exceeding 16 nodes with current methods (Blumenthal and Gamper, 2020).

2.4 Neural Network-based Graph Similarity Computation

Among early neural network-based approaches, GCNMax and GCNMin utilized the GCN architecture to generate graph-level embeddings for similarity computation (Riba et al., 2018), largely overlooking intrinsic node-level features. SMPNN considered only pairwise node similarity scores and aggregated them via simple summation (Berretti et al., 2001), failing to capture richer structural interactions.

In more recent work, SimGNN pioneered framing graph similarity computation as a regression problem (Bai et al., 2019). It processes graphs at both the node-level and graph-level: extracting node-level features via histograms and graph-level features using a NTN, with a final fully connected layer predicting similarity. However, histogram features are non-differentiable, and the model neglects subgraph level structures. Building on this, GraphSim incorporated multi-scale convolutional operations (Bai et al., 2020), using outputs from different GCN layers as graph representations, concatenating them, and employing CNNs to extract features. This approach, however, overemphasizes node-level feature interactions at the expense of global graph characteristics. HGMN performs hierarchical clustering on graphs and then compares them using node embeddings (Li et al., 2019), but it is highly sensitive to the distribution of graph structures. MGMN computes similarity by matching each node in one graph against all nodes in the other, mitigating structural sensitivity but consequently failing to leverage the inherent advantages of graph structure itself (Xiu et al., 2020). In the latest work, MB-GSC addresses structural considerations (Ktena et al., 2017) by proposing meta-structure matching and biased sampling for similarity prediction. However, the biased sampling heavily relies on data rich in substructures and may underperform on graphs with relatively simpler substructures.

3. Method

3.1 Problem Definition

Input two graphs G1 and G2, where G1 is defined as the source graph and G2 as the target graph, the GED between G1 and G2 is the minimum number of edit operations e required to transform G1 into G2. Here, e represents node or edge insertion, deletion, or substitution. Each edit operation is assigned a specific edit cost C(e).

$$G^{ged} = \min_{(e1,e2,\ldots ek)\in G1\rightarrow G2} \sum_{i=1}^{k} C(i) \tag{1}$$

Given two graphs G1 and G2, if there exists a common subgraph G3 such that no other common subgraph has more nodes than G3, then G3 is defined as the MCS of G1 and G2. The MCS-based distance is defined as 1 minus the number of nodes in G3 divided by the maximum number of nodes in G1 and G2.

$$G^{mcs}=1-\frac{N(G3)}{\max(N(G1),N(G2))} \tag{2}$$

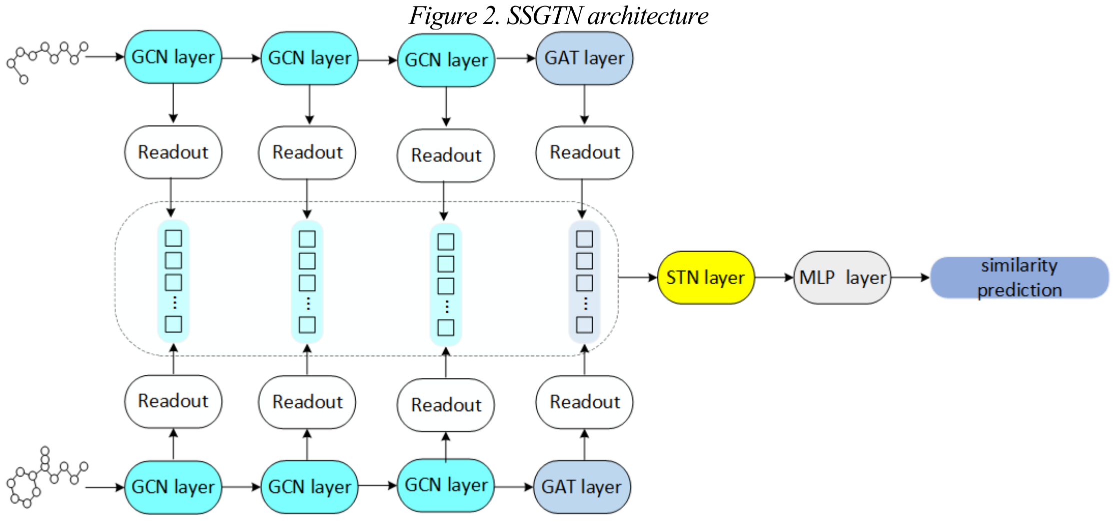
The problem addressed in this paper is graph similarity computation, which is formally defined as follows: Given a set Ç = {G_1, G_2, …, G_n}, where each element is a graph defined as $G_i = (V, E)$. Here, V = {v_1, v_2, …, v_k} represents the set of nodes in graph G_i, and E = {e_1, e_2, …, e_m} represents the set of edges in graph G_i. The graph similarity computation task is then defined as finding a function Sim() such that for any two input graphs G_1, G_2 ∈ Ç, it computes a similarity score G^{score}:

$$G^{score}= Sim(G1,G2) \tag{3}$$

3.2 Overview of the SSGTN

SSGTN takes two graphs G1 and G2 as input. The input graphs are first represented as tensors and processed by GCN to extract subgraph features. Subsequently, GAT are employed to capture global features. The model then performs two separate concatenation operations: one for subgraph features of different sizes, and another for the global graph-level features. For each concatenated subgraph feature vector, a Subgraph Tensor Network (STN) module is utilized to interact with the global feature vector, generating a fused representation that integrates both subgraph and global information. Finally, a fully connected layer maps this fused representation to the output, producing the graph similarity score. An overview of the SSGTN model architecture is illustrated in Figure 1.

Figure 2. SSGTN architecture



3.3 Subgraph and Global Feature Extraction

Before subgraph and global features can be integrated, they must first be extracted. The GCN method excels at capturing subgraph structures via neighborhood aggregation. For any input graph Gi, processing through a GCN layer yields a transformed output graph, denoted as G_{next}. Formally, the GCN operation maps the original graph Gi to this new graph representation G_{next}.

$$G_{next}= GCN(Gi) \tag{4}$$

Here, the input graph Gi is represented by its node feature matrix H and adjacency matrix A. The output graph G_{next} is correspondingly represented by a new node feature matrix H_{new}. The specific computational procedure is defined as follows:

$$H_{new}=\sigma(D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}HW) \tag{5}$$

The component formed by matrices A and D constitutes a Laplacian matrix, where I is the identity matrix and W is the learnable weight matrix. After the update, a single GCN layer can capture the features of a subgraph comprising a central node and its neighboring nodes. It follows that by stacking multiple GCN layers, the model can extract features from subgraphs of various receptive fields.

Following the extraction of subgraph features, global feature extraction is performed. While a GCN can be used to aggregate global information, the fixed, non-adaptive weighting of node contributions during feature fusion is a limitation for obtaining a high-quality global representation. To overcome this, we introduce a GAT to extract the global feature. Building upon the GCN framework, the GAT utilizes an attention mechanism to compute updated node embeddings, ultimately deriving an optimized graph-level embedding that serves as the global feature.

3.4 STN Module

To maximize the utility of both subgraph and global features, this paper proposes a STN module to effectively combine them. The STN module takes the subgraph features G_{sub} and the global feature G_{global} as input, and outputs the fused feature G_{STN}. Since G_{sub} and G_{global} often possess different dimensionalities, an STN Pooling operation is applied to project them into a shared dimensional space while striving to preserve the completeness of the subgraph features. After obtaining the dimensionally aligned subgraph feature H_{sub} and global feature H_{global}, they are directly combined as follows:

$$G_{stn}=\sigma\left(H_{sub}^{T}W_1H_{global}+W_2\begin{bmatrix}H_{sub}\\H_{global}\end{bmatrix}+b\right) \tag{6}$$

Here, W_1, W_2, and b are three learnable parameters. First, a relational tensor is extracted by modeling the interactions between the subgraph feature H_{sub} and the global feature H_{global}. Subsequently, these two features are concatenated and processed to derive a combined tensor. Finally, the relational tensor and the combined tensor are integrated to produce the final output of the module.

3.5 Graph Similarity Computation

To holistically incorporate all subgraph features and the global feature, the subgraph features of different sizes are each combined with the global feature through the STN module. The resulting set of feature vectors are then concatenated and passed through a fully connected network to produce the final similarity score.

In summary, once the model outputs a similarity score, the model is trained by minimizing the Mean Squared Error (MSE) loss between the prediction and the ground truth:

$$L_{mse}=\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y})^2 \tag{7}$$

Where y_i represents the ground-truth similarity, and $\hat{y}$ denotes the predicted similarity from the model.

4. Experiments

4.1 Datasets

In our experiments, we utilize two public, real-world datasets: LINUX and IMDBMulti. The ground-truth labels for training are the pairwise GED and Maximum MCS values between graphs within each dataset.

The LINUX dataset contains program dependency graphs derived from the Linux kernel. In these graphs, nodes represent individual code statements, and edges denote dependency relationships between them.

The IMDBMulti dataset consists of actor collaboration networks. Here, nodes represent actors/actresses, and edges indicate a co-acting relationship between them in at least one movie.

All datasets are partitioned into training, testing, and validation sets with a ratio of 6:2:2. Key statistics of the selected datasets, including the number of graphs and nodes, are summarized in Table 1.

Table 1 dataset

dataset	graph number	node number	avg node number
LINUX	1000	[4,10]	7.6
IMDBMulti	1500	[7,89]	13.0

4.2 Experimental Setup and Baseline Models

4.2.1 Experimental Environment

The experiments were conducted on a remote server with the following specifications: Ubuntu 18.04 operating system, CUDA 11.1, an NVIDIA RTX 2080 Ti GPU (11GB VRAM), and a 12 vCPU Intel(R) Xeon(R) Platinum 8255C CPU @ 2.50GHz. The Python version was 3.8, and the deep learning framework used was PyTorch version 1.8.1.

4.2.2 Parameter Settings

In our experiments, the number of GCN layers was set to 3, as this depth is sufficient to capture most subgraph features. A single GAT layer follows the GCN stacks to extract global features. The output dimensions for the three GCN layers were set to 128, 64, and 32, respectively. The GAT output dimension was set to 32, and the STN module output dimension was also 32. The fully connected network consists of 5 layers with dimensions 128, 64, 32, 16, and 1, sequentially. The Adam optimizer was used with the number of epochs set to 1500, a learning rate of 0.001, and a batch size of 128.

4.2.3 Baseline Models

To thoroughly demonstrate the effectiveness of our proposed model, we compare it against seven state-of-the-art models as baselines:

(1) GCNMean/GCNMax: These models use GCNs for graph embedding, followed by mean/max pooling, and then a fully connected layer to produce the output.

(2) SimGNN: This model considers both node-level and graph-level features. It uses histogram-based methods for node-level features and a Neural Tensor Network for graph-level features, followed by a fully connected layer.

(3) GraphSim: An extension of SimGNN, it employs CNNs to process embeddings from different scales.

(4) GMN: This model uses a variant of message-passing networks with cross-graph attention, allowing the neighborhood information from one graph to influence the node embeddings of another.

(5) MGMN: It captures cross-level interactions between nodes and graphs by computing similarity between a node in one graph and all nodes in the other graph.

(6) H2MN: This model learns graph representations from a hypergraph perspective, matching hyperedges as subgraphs to compute subgraph similarity.

4.2.4 Evaluation Metrics

Three representative evaluation metrics are used to assess model performance: Mean Squared Error (MSE), Spearman's rank correlation coefficient (ρ), and top-10 accuracy (P@10). MSE measures the average squared difference between predicted and ground-truth values. Spearman's ρ evaluates the monotonic relationship between the predicted rankings and the true rankings. P@10 is the accuracy of the top-10 nearest neighbors retrieved based on the predicted similarity.

4.3 Comparative Experimental Results

To validate the effectiveness of our model, we conducted experiments using both GED and MCS values as labels across the four datasets. The experimental results are presented in Table 2 and Table 3.

Table 2 Result of GED prediction

		LINUX			IMDB	
Model	mse	ρ	p@10	mse	ρ	p@10
GCNMean	7.541	0.579	0.141	68.823	0.402	0.219
GCNMax	6.341	0.724	0.541	58.425	0.449	0.437
SimGNN	2.360	0.943	0.775	2.964	0.781	0.724
GraphSim	1.076	0.972	0.869	1.924	0.825	0.813
GMN	2.676	0.802	0.862	3.210	0.725	0.751
MGMN	5.259	0.915	0.648	3.145	0.531	0.521
H^2MN	1.561	0.566	0.849	2.232	0.691	0.498
SSGTN	**1.001**	**0.989**	**0.941**	**1.177**	0.889	**0.829**

Table 3 Result of MCS prediction

		LINUX			IMDB	
Model	mse	ρ	p@10	mse	ρ	p@10
GCNMean	2.689	0.521	0.421	10.457	0.746	0.387
GCNMax	2.170	0.714	0.459	20.235	0.841	0.451
SimGNN	0.729	0.859	0.850	2.451	0.930	0.812
GraphSim	3.164	0.962	0.951	1.287	0.976	0.882
GMN	0.794	0.939	0.949	0.590	0.941	0.875
MGMN	0.739	0.637	0.570	5.128	0.752	0.695
H^2MN	1.541	0.762	0.637	3.176	0.739	0.603
SSGTN	**0.144**	**0.976**	0.853	**0.182**	**0.987**	**0.891**

The experimental results demonstrate that our model achieves performance that significantly surpasses current state-of-the-art models on both the LINUX and IMDBMulti datasets. The LINUX dataset features simpler structures and a smaller number of nodes, whereas IMDBMulti exhibits complex structures and a large number of nodes. Our model's superior performance across both datasets indicates its capability to extract sufficient features without relying exclusively on complex subgraph structures. Simultaneously, when confronted with data possessing intricate structures, it effectively leverages the abundant features provided by the subgraph information. This robust performance validates the effectiveness of our proposed model.

5. Conclusion

To address certain limitations in existing graph similarity computation methods, this paper proposed the SSGTN model. By extracting and effectively integrating both subgraph and global features, our model achieves more accurate graph similarity computation. This approach successfully leverages the rich information embedded in local subgraph structures while maintaining a strong focus on the global graph characteristics, leading to higher accuracy. Extensive experimental results demonstrate that the proposed model effectively improves the accuracy of graph similarity computation.

References

An, L., Wu, A., Yuan, Y., SUN, S. Q. and WANG, G. R., (2023). Graph similarity computation method based on meta-structures matching and biased sampling. *Chinese Journal of Computers,* vol. 46**,** no. 7, pp. 1513-1531.

Bai, Y., Ding, H., Bian, S., Chen, T., Sun, Y. and Wang, W., (2019). Published. Simgnn: A neural network approach to fast graph similarity computation. Proceedings of the twelfth ACM international conference on web search and data mining, Melbourne, VIC. ACM, pp. 384-392.

Bai, Y., Ding, H., Gu, K., Sun, Y. and Wang, W., (2020). Published. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. Proceedings of the AAAI conference on artificial intelligence, New York, NY. AAAI, pp. 3219-3226.

Berahmand, K., Saberi-Movahed, F., Sheikhpour, R., Li, Y. and Jalili, M., (2025). A comprehensive survey on spectral clustering with graph structure learning. *arXiv preprint arXiv:2501.13597.*

Berretti, S., Del Bimbo, A. and Vicario, E., (2001). Efficient matching and indexing of graph models in content-based retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 23, no. 10, pp. 1089-1105.

Blumenthal, D. B. and Gamper, J., (2020). On the exact computation of the graph edit distance. *Pattern Recognition Letters,* vol. 134, pp. 46-57.

Bunke, H., (1997). On a relation between graph edit distance and maximum common subgraph. *Pattern recognition letters,* vol. 18, no. 8, pp. 689-694.

Bunke, H. and Shearer, K., (1998). A graph distance metric based on the maximal common subgraph. *Pattern recognition letters,* vol. 19, no. 3-4, pp. 255-259.

Bunke., H., (1983). What is the distance between graphs. *Bulletin of the EATCS 20,* vol. 20, no. 1983, pp. 35-39.

Kipf, T., (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907.*

Koutra, D., Vogelstein, J. T. and Faloutsos, C., (2013). Published. Deltacon: A principled massive-graph similarity function. Proceedings of the 2013 SIAM international conference on data mining, Austin, Texas. SIAM, pp. 162-170.

Ktena, S. I., Parisot, S., Ferrante, E., Rajchl, M., Lee, M., Glocker, B. and Rueckert, D., (2017). Published. Distance Metric Learning Using Graph Convolutional Networks: Application to Functional Brain Networks. Medical Image Computing and Computer Assisted Intervention − MICCAI 2017, Cham. Springer International Publishing, pp. 469-477.

Li, Y., Gu, C., Dullien, T., Vinyals, O. and Kohli, P., (2019). Published. Graph matching networks for learning the similarity of graph structured objects. International conference on machine learning, Long Beach. PMLR, pp. 3835-3845.

Ling, X., Wu, L., Wang, S., Ma, T., Xu, F., Liu, A. X., Wu, C. and Ji, S., (2021). Multilevel graph matching networks for deep graph similarity learning. *IEEE Transactions on Neural Networks and Learning Systems,* vol. 34, no. 2, pp. 799-813.

Liu, J., Ma, G., Jiang, F., Lu, C. T., Yu, P. S. and Ragin, A. B., (2019). Published. Community-preserving Graph Convolutions for Structural and Functional Joint Embedding of Brain Networks. 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA. IEEE, pp. 1163-1168.

Ma, G., Ahmed, N. K. and Willke, T. L., (2021). Deep graph similarity learning: A survey. *Data Mining and Knowledge Discovery,* vol. 35, no. 3, pp. 688-725.

Noble, C. C. and Cook, D. J., (2003). Published. Graph-based anomaly detection. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, Washington, DC. ACM, pp. 631-636.

Riba, P., Fischer, A., Lladós, J. and Fornés, A., (2018). Published. Learning graph distances with message passing neural networks. 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China. IEEE, pp. 2239-2244.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. and Bengio, Y., (2018). Published. Graph Attention Networks. International Conference on Learning Representations, Vancouver, Canada. ICLR, pp. 1-12.

Wang, L., Zheng, Y., Jin, D., Li, F., Qiao, Y. and Pan, S., (2024). Contrastive graph similarity networks. *ACM Transactions on the Web,* vol. 18, no. 2, pp. 1-20.

Xiu, H., Yan, X., Wang, X., Cheng, J. and Cao, L., (2020). Hierarchical graph matching network for graph similarity computation. *arXiv preprint arXiv:2006.16551.*

Zeng, Z., Tung, A. K., Wang, J., Feng, J. and Zhou, L., (2009). Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment,* vol. 2, no. 1, pp. 25-36.

Zhang, Z., Bu, J., Ester, M., Li, Z., Yao, C., Yu, Z. and Wang, C., (2021). Published. H2mn: Graph similarity learning with hierarchical hypergraph matching networks. Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, Virtual. ACM, pp. 2274-2284.

Zhuo, W. and Tan, G., (2022). Efficient graph similarity computation with alignment regularization. *Advances in Neural Information Processing Systems,* vol. 35, pp. 30181-30193.

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgment

This work was supported by the GuiZhou University of Finance and Economics 2021 Undergraduate University-level Research Project [Grant Number 730321122901].