DOI: 10.19884/j.1672-5220.202409006

Subgraph Matching on Multi-Attributed Graphs Based on Contrastive Learning

LIU Bozhi, FANG Xiu, SUN Guohao*, LU Jinhu
School of Computer Science and Technology, Donghua University, Shanghai 201620, China

Abstract: Graphs have been widely used in fields ranging from chemical informatics to social network analysis. Graph-related problems become increasingly significant, with subgraph matching standing out as one of the most challenging tasks. The goal of subgraph matching is to find all subgraphs in the data graph that are isomorphic to the query graph. Traditional methods mostly rely on search strategies with high computational complexity and are hard to apply to large-scale real datasets. With the advent of graph neural networks (GNNs), researchers have turned to GNNs to address subgraph matching problems. However, the multi-attributed features on nodes and edges are overlooked during the learning of graphs, which causes inaccurate results in real-world scenarios. To tackle this problem, we propose a novel model called subgraph matching on multi-attributed graph network (SGMAN). SGMAN first utilizes improved line graphs to capture node and edge features. Then, SGMAN integrates GNN and contrastive learning (CL) to derive graph representation embeddings and calculate the matching matrix to represent the matching results. We conduct experiments on public datasets, and the results affirm the superior performance of our model.

Keywords: subgraph matching; graph neural network (GNN); multi-attributed graph; contrastive learning (CL)
CLC number: TP389.1 **Document code**: A
Article ID: 1672-5220(2025)05-0523-11

Open Science Identity (OSID)

0 Introduction

In recent years, there has been extensive research on graph-related problems, with applications ranging from chemical informatics to social network analysis. Graph-related problems become increasingly significant, with subgraph matching standing out as one of the most challenging tasks. Subgraph matching aims to find all subgraphs in the data graph that are isomorphic to the query graph. Figure 1 illustrates a subgraph matching application in social network analysis. Consider a scenario where a work group requires its member A to be acquainted with both individuals B and C, where A, B and C can represent labels for family members, jobs, geographic locations, etc. We can formulate a query graph and utilize a subgraph matching algorithm to search for the groups meeting the specified conditions in a known extensive social network graph. As shown in Fig. 1, the algorithm finds two matching subgraphs: one consisting of A_1, B_1 and C, and another consisting of A_2, B_2 and C. E and F are other unrelated nodes in the data graph.

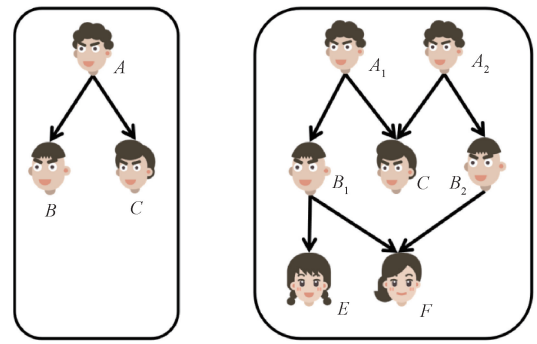

Fig. 1 Subgraph matching applied in social network analysis

Subgraph matching is a challenging NP-complete problem. Traditional methods typically utilize search strategies to analyze the neighbor structure of nodes and sequentially match all nodes, presenting a substantial computational challenge, especially when dealing with extensive real datasets. Therefore, researchers seek more efficient alternatives. The advent of graph neural networks (GNNs) has spurred the development of several GNN-based methods for subgraph matching, including Sub-GMN[1], AEDNet[2] and DMPNN[3]. These methods derive embedding representations by learning graph features, followed by predicting the matching matrix. The robust learning capabilities of GNNs ensure accurate predictions, while the parallel processing prowess of GPUs ensures efficient computations.

While existing GNN-based methods have made some progress in subgraph matching, they focus mainly on basic structural matching. However, real-world graphs are intricate, with multiple attributes on nodes and edges. Figure 2 provides an example of subgraph matching on multi-attributed graphs, displaying nodes with diverse attribute details and multiple interconnections among them. Considering both attribute and structure

Received date: 2024-09-18
* Correspondence should be addressed to SUN Guohao, email: ghsun@dhu.edu.cn
Citation: LIU B Z, FANG X, SUN G H, et al. Subgraph matching on multi-attributed graphs based on contrastive learning[J]. *Journal of Donghua University (English Edition)*, 2025, 42(5): 523-533.

information, the matching results will be the subgraph consisting of A_1, B_1 and C, while the subgraph consisting of A_2, B_2 and C can not match the query graph. The existing GNN-based methods focus only on structural matching, and in this case, the subgraph consisting of A_2, B_2 and C will appear in their matching results. Hence, innovative approaches are required to address subgraph matching on multi-attributed graphs.

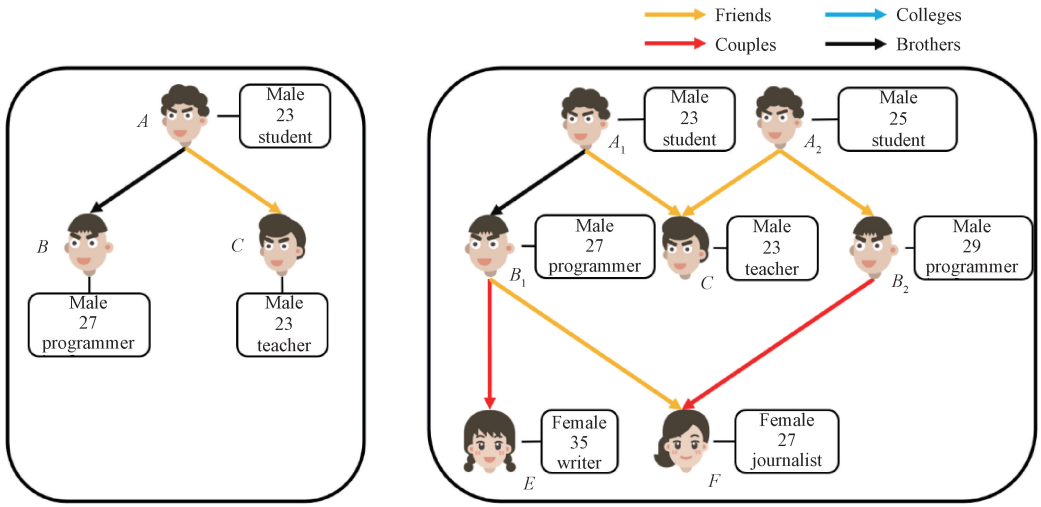

Fig. 2 Example of subgraph matching on multi-attributed graphs

The concept of line graph was first proposed in the graph theory[4] to address various graph theoretic problems in mathematical modeling, including the shortest path and the minimum spanning tree problems. In a line graph, edges are transformed into vertices, representing the adjacency of edges in the original graph. In recent years, researchers have utilized the line graph to enrich the learned features of GNNs, enabling them to tackle diverse graph-related tasks such as node classification[5] and link prediction[6]. We realize that the line graph offers a convenient way to acquire abstract edge information, addressing previous limitations of ignoring the attribute information on the nodes and edges when matching subgraphs. In our proposed model, we adopt the line graph to acquire attribute information. Contrastive learning (CL)[7] has been applied in many scenarios, such as computer vision (CV)[8] and graph data mining[9]. CL effectively improves the embedding quality by minimizing the distance between positive pairs in the embedding space. As a typical learning method, CL can mine the intrinsic features of data to improve the downstream task[10]. We recognize that subgraph matching inherently involves contrastive relationships between nodes (matching nodes and mismatching nodes). Utilizing these contrastive relationships can improve the quality of the embeddings learned by GNNs, thereby improving the matching accuracy. Existing methods tend to ignore these contrastive relationships contained in subgraph matching, which can lead to less robust model prediction performance. Therefore, in our proposed model, we apply CL to improve the quality of the embeddings learned by GNNs.

While the line graph can acquire attribute information, it may lead to information loss and an increase in graph complexity, affecting the learning of graphs. Therefore, the first challenge in our method is how to enhance the line graph to avoid information loss and an increase in graph complexity. In addition, the key to CL is generating appropriate anchors, positive and negative instances. Inappropriate positive and negative instances will directly lead to low-quality embeddings. Hence, the second challenge is how to construct a CL framework to select appropriate contrastive pairs and improve the quality of embeddings. To address these challenges, our model adds reverse edges when transforming the line graph and utilizes the matching relationship between nodes in the query graph and data graph.

Overall, we summarize the main contributions of this research as follows.

1) To obtain complete and accurate features on edges, we propose an improved line graph. The new line graph adds reverse edges to those edges in the original graph. This ensures the converted line graph retains complete information, allowing our model to capture comprehensive edge features while maintaining the low complexity of the line graph structure.

2) To improve the quality of embeddings, we construct a CL framework in our model. To our knowledge, it is the first to incorporate CL into subgraph matching.

3) We conduct experiments on three public datasets, and the experimental results show that our model outperforms existing models, proving the effectiveness of our model.

In the first section of this paper, the related work on

subgraph matching is presented; in the second section, problem definition and preliminaries are described; in the third section, the methodology and the model proposed in this paper are presented; in the fourth section, the experimental results are presented.

1 Related Work

The existing methods to deal with subgraph matching problems can be divided into two categories: traditional search-based methods and GNN-based methods. We review the related work based on these two categories.

1.1 Traditional search-based methods

Traditional methods mainly employ a search strategy to analyze the neighbor structure of nodes and match all nodes sequentially. The first subgraph isomorphism algorithm proposed by Ullmann[11] uses a depth-first search strategy to enumerate subgraphs in the data graph that match the query graph. VF[12] makes improvements in the matching order and pruning strategy. Since then, many improved algorithms have been proposed that consider more complex substructure features to prune the results, thereby speeding up the search process. For example, GraphQL[13] utilizes a depth-first search spanning tree to filter candidate nodes. These algorithms enhance candidate node filtering from various perspectives, effectively reducing the frequency of backtracking during the search process. In addition, some index-based matching algorithms have been proposed. Among them, GraphGrep[14] is a typical path-based indexing algorithm that enumerates paths in the graph with lengths lower than a predefined value as indexed features and uses the indices to reduce the search space. To achieve distributed processing, the approach in Ref. [15] decomposes the query graph based on node degrees, thus leveraging the resulting subqueries as an index. By reusing indexing information, they significantly reduce the indexing load, which becomes crucial in scenarios where multiple queries are issued simultaneously. Most of the methods proposed in recent years focus on filtering candidate sets. G-finder[16] employs two key techniques: a novel auxiliary data structure for efficiently indexing candidate vertices, and a dynamic filtering and refinement strategy to early prune false candidates. CECI[17] introduces three unique techniques: BFS-based filtering and reverse-BFS-based refinement for early pruning of unpromising candidates; set intersection for faster candidate verification; search cardinality-based cost estimation for proactive detection and division of large embedding clusters. GuP[18] proposes a subgraph matching algorithm with pruning based on guards. By attaching a guard to each candidate vertex and edge, GuP adaptively filters out unnecessary candidates depending on the search state at each step. MGSM[19] presents a new matching algorithm called Tps, which is noted for its strategic vertex searching order. To achieve efficient graph indexing, a graph indexing scheme is introduced in Ref. [20], which operates by monotonically counting path and cycle features under relaxed semantics.

All of the traditional search-based methods face the common problem that they have very high computational complexity and are hard to apply to large-scale datasets. What's more, they do not consider the attribute information in the graph, which means that they cannot be applied to the subgraph matching problem on multi-attributed graphs.

1.2 GNN-based methods

With the rising popularity of GNNs, researchers have advocated for using neural network models to address subgraph matching problems. Sub-GMN directly forces node-level embeddings of two matched nodes to be close to each other and makes the corresponding node-level embeddings similar by combining GCN[21] and NTN[22]. NeuroMatch[23] decomposes query and data graphs into small subgraphs, training the model to capture geometric constraints corresponding to subgraph relations, and then performing subgraph matching directly in the embedding space. AEDNet[2] proposes an adaptive edge-deleting mechanism to remove extra edges to ensure consistency of the adjacency structure of matched nodes. DMPNN[3] starts from a particular edge-to-vertex transform and exploits the isomorphism property in the edge-to-vertex dual graphs.

In other graph-related scenarios, researchers have developed models to extract features from multi-attributed graphs. GATNE[24] divides the node embedding into two parts to learn different types of information, respectively, and finally combines them to obtain node representations containing comprehensive information. For graphs containing different edge relationships, RGCN[25] employs a dedicated aggregation strategy for each edge type, applying distinct transformations based on the type of edge. CompGCN[26] leverages a variety of composition operations from knowledge graph embedding techniques[27] to jointly embed both nodes and relations in a graph.

Compared with the traditional search-based methods, the existing GNN-based methods can achieve better matching efficiency because of the robust learning of GNNs and the parallel processing ability of GPUs. The ways of learning feature information lead to differences in the performance of subgraph matching accuracy. In our model, we apply the CL to learn feature information and improve the quality of the embeddings, thereby improving the subgraph matching accuracy.

2 Problem Definition and Preliminaries

2.1 Problem definition

A graph is denoted as a tuple $\{V, E, F^{n}, F^{e}\}$, where V is the node set, E is the edge set, and F^{n} and F^{e} are feature functions that map each node and each edge to feature vectors, respectively. In this paper, $Q = \{V_Q, E_Q, F_Q^{n}, F_Q^{e}\}$ represents a query graph, and $G = \{V_G, E_G, F_G^{n}, F_G^{e}\}$ stands for a data graph.

The subgraph matching on multi-attributed graphs is an injection function $m: V_Q \rightarrow V_G$ which satisfies:

1) $\forall u \in V_Q$, $m(u) \in V_G$ and $F^n(u) = F^n(m(u))$;

2) $\forall (u_a, u_b) \in E_Q$, $(m(u_a), m(u_b)) \in E_G$ and $F^e(u_a, u_b) = F^e(m(u_a), m(u_b))$.

Here, u denotes a node; u_a and u_b represent two distinct nodes. There may be multiple mappings from V_Q to V_G. We use $M(Q, G) = \{m_1, m_2, \cdots, m_k\}$ for the set of all mappings. The subgraph matching problem is to find the set of all mappings $M(Q, G)$ for a given pair of graphs.

2.2 Line graph

The line graph H of a data graph G represents the adjacencies between edges of G with an edge-to-vertex map denoted as $g: E_G \rightarrow V_H$.

A line graph H of a data graph G is obtained by associating a vertex $v' \in V_H$ with each edge $e \in E_G$ and connecting two vertices $u', v' \in V_H$ with an edge from u' to v' if and only if the destination of the corresponding edge d in the data graph G is exactly the source of e. Formally, we have the following conclusions.

1) $\forall e = (u, v) \in E_G, Y_G(e) = X_H(g(e))$;

2) $\forall v' \in V_H, X_H(v') = Y_G(g^{-1}(v'))$;

3) $\forall d, e \in E_G, u' = g(d) \in V_H, v' = g(e) \in V_H$;

4) $\forall e' = (u', v') \in E_H, d = g^{-1}(u') \in E_G, e = g^{-1}(v') \in E_G$.

Here, e and d denote different edges; v denotes a node; Y_G denotes the edge feature function of the data graph G; X_H denotes the node feature function of the line graph H. Figure 3 demonstrates the process of converting an original graph into a line graph. In the line graph, each node denotes an edge of the original graph. If the destination of an edge connects to the source of another edge in the original graph, an edge is formed in the line graph, corresponding to node b in the original graph. This process continues for the remaining edges in the line graph. Undirected graphs can be considered as special cases of bidirectional graphs.

However, this type of line graph may pose some problems, such as the loss of information for nodes a and c as depicted in the original graph. In this paper, we propose a new way of line graph construction to avoid such problems.

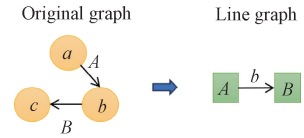

Fig. 3 Example of converting an original graph to a line graph

3 Methodology

3.1 Overview

Our proposed model, subgraph matching on multi-attributed graph network (SGMAN), mainly consists of four modules: a graph transform module, a message passing module, an embedding contrast module, and a result prediction module. Figure 4 shows an overview of SGMAN. Firstly, in the graph transform stage, the original graphs convert into line graphs. Secondly, in the message passing stage, input original graphs and line graphs, employing convolutional operations to learn graph features (where h and z represent embeddings). Thirdly, in the embedding contrast stage, we utilize the contrastive relationships between nodes to construct contrastive pairs, aiming to improve the quality of embeddings by minimizing the distance between anchors and positives and maximizing the distance between anchors and negatives. Finally, in the result prediction stage, compute the matching matrix as a prediction.

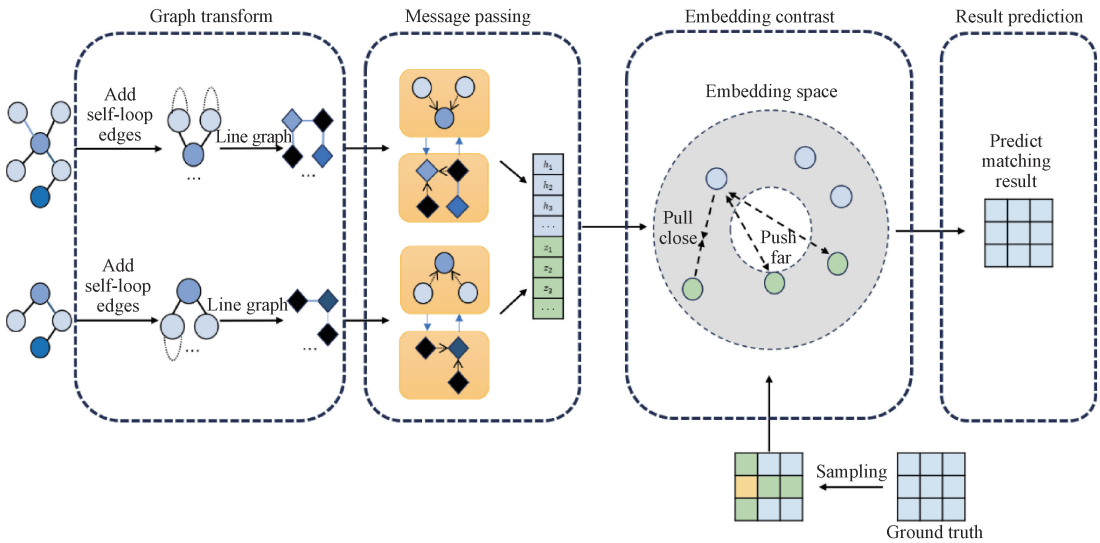

Fig. 4 Model overview of SGMAN

3.2 Graph transform

In the graph transform stage, we first convert the original graphs to line graphs. However, as mentioned in subsection 2.2, the way of constructing line graphs will lead to the problem of information loss, and in the subgraph matching process, such information loss will result in two non-isomorphic graphs having isomorphic line graphs.

Figure 5 is an example of problems caused by line graphs. The original graph 1 and original graph 2 are two non-isomorphic graphs, but they have isomorphic line graphs. We identify that the problem arises from the omission of nodes with a degree of 1 in the original graph during the conversion to a line graph. Neglecting this situation and directly employing line graphs to capture node and edge features will result in the model learning inaccurate features, thereby leading to poor performance in subgraph matching tasks.

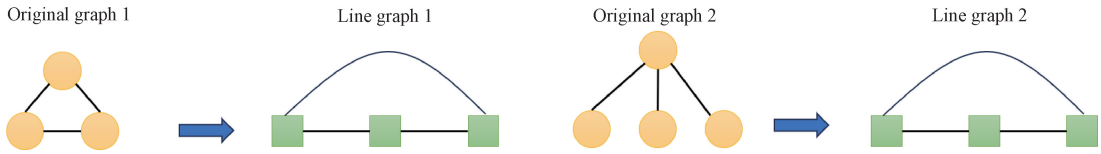

Fig. 5 Example of problems caused by line graphs

DMPNN realizes this problem and proposes to add reverse edges to each edge of the original graph to prevent information loss in the line graph. However, this method introduces significant redundancy. As shown in Fig. 6(a), there are four edges b in the line graph, and such redundancy will limit the performance of graph learning tasks. To deal with the above-mentioned problems, we propose a new way to construct a line graph. First, we add self-loop edges to nodes with a degree of 1. This operation ensures that all nodes have a degree higher than 1. Then, we convert the new graph to the improved line graph. Figure 6(b) is an example of our way. The advantages of our improved line graph can be summarized below.

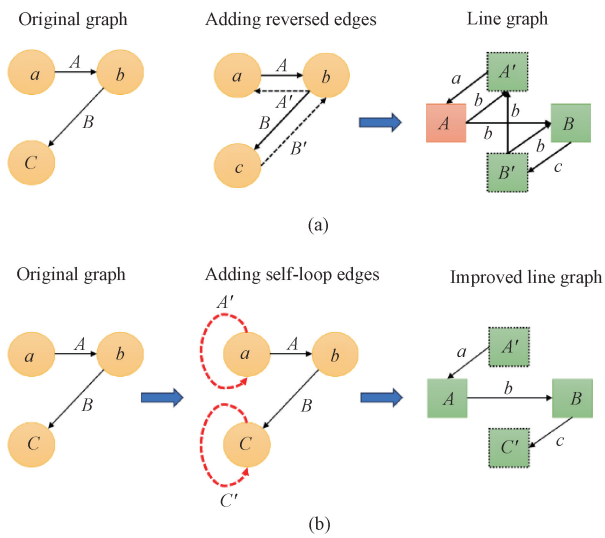

Fig. 6 Comparison of line graphs construction between DMPNN and our method: (a) construction of line graph in DMPNN; (b) construction of line graph in our method

1) Preserve the integrity of information. Our improved line graph avoids the loss of node information by introducing self-loop edges, ensuring that nodes with a degree of 1 are retained, thus maintaining the completeness of graph information.

2) Reduce graph complexity. Compared to DMPNN, our improved line graph significantly reduce graph complexity and eliminates redundant information. This facilitates the following graph learning process, enhancing overall efficiency.

3.3 Message passing

After obtaining the improved line graph, we apply convolution to learn the graph features in the message passing stage. Figure 7 is a description of this stage. In the original graph, a node obtains information from its neighbor nodes, and at the same time, obtains information from the corresponding nodes in the line graph.

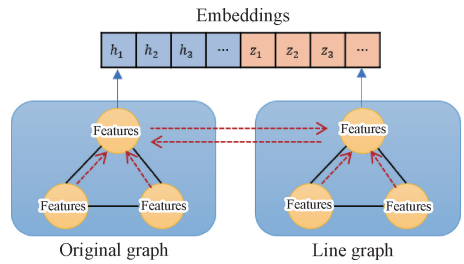

Fig. 7 Schematic of message passing process between original graph and line graph

In our model, the edge embeddings in the original graph are the node embeddings in the corresponding line graph. To ensure that nodes in the original graph capture features from edges, we combine the embeddings of nodes in the line graph with the embeddings of nodes in the original graph as the new embeddings in the next layer. We then apply similar operations in the line graph, allowing its nodes to learn features from the original graph. By doing so, nodes can aggregate new information from neighbor structures in the next layer.

The convolution equation is given as

$$\boldsymbol{h}^{l+1} = (\gamma_0 + \gamma_1)\boldsymbol{h}^l + \frac{2\gamma_1}{\lambda}(\boldsymbol{D} + \boldsymbol{I}_m)\boldsymbol{h}^l + \frac{\gamma_1}{\lambda}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{z}^l, \quad (1)$$

where $\boldsymbol{D}$ is the degree matrix; $\boldsymbol{I}_m$ is the identity matrix; $\boldsymbol{B}$ is the oriented incidence matrix; $\boldsymbol{h}^l$ is the feature vector

corresponding to nodes in the original graph in the lth layer; z^l is the feature vector corresponding to nodes in the line graph in the l th layer; γ denotes the trainable parameter; λ is the scaling parameter.

3.4 Embedding contrast

After message passing, in the embedding contrast module, we utilize the contrastive relationships of nodes to improve the quality of embedding by minimizing the distance of anchors and positives while simultaneously maximizing the distance of anchors and negatives in the embedding space.

We employ the exact subgraph matching algorithm VF3[28] to generate the binary ground truth, denoted as g, for the query graph with j nodes and the data graph with k nodes, and $g[j][k] = 1$ means a match for the jth node in the query graph and the kth node in the data graph. We set the kth node in the data graph with $g[j][k] = 1$ as the positive instance for the jth node in the query graph, and the remaining nodes in the data graph with $g[j][k] = 0$ as negative instances.

It is worth noting that a query graph may match multiple matching subgraphs in the data graph, and therefore, one node in a query graph can have multiple matching nodes in a data graph. In such cases, one node will have multiple positive instances. Therefore, different from self-supervised CL, the contrastive loss in our method will consider multiple positive instances. The formula for calculating the contrastive loss L is

$$L = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \frac{\exp(\boldsymbol{h}_i \cdot \boldsymbol{h}_p / \tau)}{\sum_{a \in A(i)} \exp(\boldsymbol{h}_i \cdot \boldsymbol{h}_a / \tau)}, \quad (2)$$

where the index i is the anchor; I contains all data in a batch; $P(i)$ is the set of indices of all positives in the batch, the other indices are negatives, and $|P(i)|$ is its cardinality; $A(i)$ represents $I/\{i\}$; τ is a scalar temperature parameter; $\boldsymbol{h}$ denotes embedding vectors in different layers; $\cdot$ denotes the inner (dot) product. The purpose of this loss function is to minimize the distance between the matching nodes in the embedding space while maximizing the distance between mismatching nodes. Calculating the loss in this way offers the following advantages.

1) Attractive power with multiple positives. We utilize the matching relationship between nodes to construct multiple positive instances. The loss encourages the nodes and positives to close in the embedding space, resulting in a better quality of the embedding.

2) Contrastive power with many negatives. This property is crucial for representation learning using CL, as several studies have demonstrated improved performance with an increased number of negatives. Therefore, we regard all mismatching nodes as negatives to enhance the performance of CL.

3.5 Result prediction

In the result prediction stage, the model outputs the prediction results for subgraph matching. Following the preceding steps, we obtain embeddings of nodes. We calculate the similarity between the embeddings of the nodes in the query graph and the data graph as the similarity matrix. This matrix reflects the matching results of specific nodes between the two graphs, serving as the output of the model.

4 Experiments

We conduct extensive experiments on three publicly available datasets and compare the performance with state-of-the-art GNN-based methods to validate our proposed model. Our goal is to answer the following questions.

Q1: How does SGMAN perform compared to other GNN-based methods?

Q2: How do different parts of SGMAN contribute to the final performance?

Q3: How do different parameter settings affect the performance of SGMAN?

Q4: How does the performance improve when incorporating the proposed components into other graph models?

4.1 Experimental setup

In this section, we introduce the datasets used in the experiment, the baselines for comparison, the evaluation metrics and the implementation details.

4.1.1 Datasets

Tables 1 and 2 show the statistics of three datasets: the dataset Regular, which uses 3-stars, triangles, tailed triangles and chordal cycles as patterns; the mutagenic compound dataset Mutag with 24 patterns; the heterogeneous dataset Complex with 75 random patterns. Train, valid and test denote the number of graphs in the corresponding dataset, respectively. DMPNN provides these datasets, and to ensure a fair comparison, we follow the same processing method as DMPNN for dividing the train, valid and test sets in the dataset, and the details are shown in Table 1. In Table 2, $|p|$ and $|g|$ correspond to query graphs and data graphs, $|V|$ is the number of vertexes, $|E|$ is the number of edges, $|X|$ is the number of labeled types of nodes, $|Y|$ is the number of labeled types of edges. Max and avg represent the maximum and average values of the number of nodes in all graphs in the dataset, respectively.

Table 1 Distribution of graphs across dataset

Dataset	Train	Valid	Test
Regular	6 000	4 000	10 000
Mutag	1 400	1 500	1 500
Complex	358 000	44 000	44 000

Table 2 Statistics of graph features

Metric	Regular		Mutag		Complex	
	Max	Avg	Max	Avg	Max	Avg
$\lvert V_p \rvert$	4.0	3.8	4.0	3.5	8.0	5.2
$\lvert E_p \rvert$	10.0	7.5	3.0	2.5	8.0	5.9
$\lvert X_p \rvert$	1.0	1.0	2.0	1.5	8.0	3.4
$\lvert Y_p \rvert$	1.0	1.0	2.0	1.5	8.0	3.8
$\lvert V_g \rvert$	30.0	18.8	28.0	17.9	64.0	32.6
$\lvert E_g \rvert$	90.0	62.7	66.0	39.6	256.0	73.6
$\lvert X_g \rvert$	1.0	1.0	16.0	9.0	7.0	3.3
$\lvert Y_g \rvert$	1.0	1.0	16.0	9.4	4.0	3.0

4.1.2 Baselines

We select the competitive network models capable of handling multi-relational graphs as baselines, including TXL, RGCN, RGIN, CompGCN and DMPNN. TXL is a 6-layer transformer encoder with additional memory. RGCN is a 3-layer RGCN with the block-diagonal decomposition. We follow the same setting in DMPNN to use mean-pooling in the message propagation part. RGIN combines GIN and RGCN: adding a 2-layer MLP[29] after each relational convolutional layer and using the sum aggregator. These models are versatile for learning multi-relational graph data. We use the obtained graph representations to calculate the matching matrix, which is then compared with the ground truth obtained from the VF3 exact matching algorithm.

4.1.3 Evaluation metrics

Our main focus is to evaluate the performance of different graph models in predicting subgraph matching. We consider subgraph matching as a regression problem and compare the prediction results with the ground truth, calculating the mean absolute error (MAE), root mean square error (RMSE)[30] and accuracy (ACC). MAE represents the average of the absolute errors of the predicted and actual matching results. RMSE represents the sample standard deviation of the difference between the correctly predicted and actual matching results. ACC represents the accuracy of the matching results. Lower MAE and RMSE values indicate better performance, while higher ACC values signify improved accuracy.

4.1.4 Implementation details

We first perform a complexity analysis. In the phase of converting the graph into a line graph, the complexity is $O(n+e)$. In the phase of message passing and embedding contrast, the complexity is $O(n)$.

To ensure a fair comparison, we maintain the uniformity in embedding dimensions, hidden sizes and filter numbers, setting them all to 64. The segment size and memory size in TXL are 128, based on the computational complexity consideration. We use MSE to train the model and select the best model based on the results of the development set. The optimizer is AdamW[31] with a base learning rate of 1×10^{-3} and a weight decay coefficient of 1×10^{-5}. LeakyReLU serves as the activation function across all modules. Training and evaluations of models are done on an NVIDIA 4090 GPU under the PyTorch and DGL frameworks.

4.2 Overall performance (Q1)

We compare the performance of SGMAN with baselines. Table 3 shows the performance of each model on the three datasets. The best performance is shown in bold. Imp represents the improvement in performance between SGMAN and other models; w/o represents without in the ablation experiment.

Table 3 Overall performance comparison

Model	Regular			Mutag			Complex		
	MAE	RMSE	ACC	MAE	RMSE	ACC	MAE	RMSE	ACC
TXL	10.721	15.263	0.819	0.830	1.895	0.912	9.576	43.055	0.762
RGCN	12.676	15.833	0.847	0.227	1.043	0.947	8.939	27.842	0.805
RGIN	10.074	15.578	0.872	0.203	0.545	0.957	6.377	24.231	0.836
CompGCN	11.790	16.478	0.865	0.211	0.809	0.956	7.714	26.134	0.817
DMPNN	10.016	15.680	0.878	0.167	0.474	0.961	3.186	22.778	0.847
SGMAN	**8.332**	**13.229**	**0.896**	**0.046**	**0.145**	**0.992**	**2.465**	**17.634**	**0.877**
Imp	0.168	0.156	0.018	0.724	0.694	0.031	0.226	0.226	0.030
w/o CL&loop	9.944	15.566	0.858	0.191	0.440	0.958	3.982	28.392	0.801
w/o CL	8.833	13.778	0.884	0.070	0.426	0.965	2.519	20.015	0.852
w/o loop	8.736	13.234	0.887	0.079	0.431	0.963	2.848	20.237	0.849

Based on the data presented in Table 3, SGMAN outperforms other models across all evaluation metrics and datasets, highlighting the efficacy of our approach. As the complexity of graph attributes increases, SGMAN demonstrates the best performance compared to other models, particularly in Mutag and Complex datasets. This is because line graphs leverage their significant advantage on multi-attributed graphs, and combined with CL, provide higher-quality embeddings, resulting in better matching results. Even on the Regular dataset where graph attributes are simple, SGMAN performs well by capturing higher-order information, outperforming baselines.

4.3 Analysis of ablation experiments (Q2)

We conduct ablation experiments to assess the contribution of each component to the model. Our study comprises two primary components: adding self-loop edges to the original graph and transforming the original graph into a line graph; integrating CL into the model and utilizing the contrastive relationships between nodes to improve the quality of embeddings. The ablation experiment is categorized into three parts: removing only loop from the model, removing only CL, and removing both self-loop and CL. The results are presented in the last three rows of Table 3.

The performance of the model decreases when specific components are excluded, highlighting the effectiveness of our proposed method. Notably, the model exhibits a poorer performance when both loop and CL are removed. In addition, on the Mutag and Complex datasets containing multiple attributes, models without CL perform slightly better than those without line graphs. This is because line graphs offer significant advantages in capturing features on multi-attributed graphs, and the benefits of CL are built upon the rich feature information obtained from line graphs.

4.4 Parameter analyses (Q3)

We conduct experiments on the Regular and Mutag datasets to investigate how the temperature parameter τ impacts subgraph matching accuracy. The results are depicted in Fig. 8. The model achieves optimal results when τ is between 0.075 and 0.125. According to Ref. [7], a smaller temperature parameter is more beneficial for training, allowing a greater focus on learning from hard negative instances. This facilitates the separation of similar node embeddings in the embedding space. Conversely, a higher temperature parameter makes distinguishing hard negative instances from positive instances challenging, resulting in lower-quality embeddings. Therefore, selecting an appropriate temperature parameter ensures a more uniform distribution in the embedding space, leading to better results.

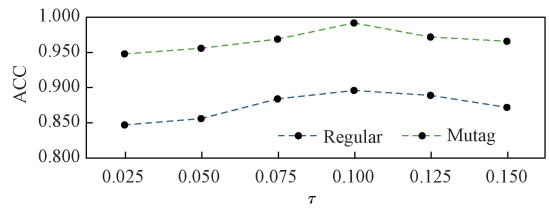

Fig. 8 Impact of temperature parameter on ACC of Mutag and Regular datasets

4.5 Performance on other models (Q4)

We assess the impact of improved line graphs and CL on baseline models, focusing our evaluation on Regular and Mutag datasets for efficiency. The results are shown in Tables 4 and 5, and w/ represents with in the ablation experiment.

Table 4 Performance comparison after introducing self-loop edges on datasets

Model		Regular			Mutag		
		MAE	RMSE	ACC	MAE	RMSE	ACC
TXL	w/o loop	10.721	15.263	0.819	0.830	1.895	0.912
	w/ loop	9.578	14.251	0.836	0.622	1.122	0.945
RGCN	w/o loop	12.676	15.833	0.847	0.227	1.043	0.947
	w/ loop	9.910	14.621	0.858	0.145	0.570	0.971
RGIN	w/o loop	10.074	15.578	0.872	0.203	0.545	0.957
	w/ loop	9.632	15.309	0.882	0.075	0.248	0.977
CompGCN	w/o loop	11.790	16.478	0.865	0.211	0.809	0.956
	w/ loop	10.364	14.568	0.876	0.106	0.440	0.976
DMPNN	w/o loop	10.016	15.680	0.878	0.167	0.474	0.961
	w/ loop	9.541	14.504	0.888	0.083	0.378	0.979

Table 5 Performance comparison after introducing CL on datasets

Model		Regular			Mutag		
		MAE	RMSE	ACC	MAE	RMSE	ACC
TXL	w/o CL	10.721	15.263	0.819	0.830	1.895	0.912
	w/ CL	9.988	14.322	0.842	0.722	1.013	0.933
RGCN	w/o CL	12.676	15.833	0.847	0.227	1.043	0.947
	w/ CL	10.697	15.378	0.857	0.210	0.556	0.961
RGIN	w/o CL	10.074	15.578	0.872	0.203	0.545	0.957
	w/ CL	9.929	15.091	0.889	0.123	0.357	0.975
CompGCN	w/o CL	11.790	16.478	0.865	0.211	0.809	0.956
	w/ CL	10.561	15.283	0.874	0.126	0.489	0.974
DMPNN	w/o CL	10.016	15.680	0.878	0.167	0.474	0.961
	w/ CL	9.975	15.204	0.892	0.121	0.341	0.970

As depicted in Table 4, introducing line graphs enhances subgraph matching accuracy and reduces matching errors across all models. Similarly, the results presented in Table 5 demonstrate a performance boost in predicting subgraph matching problems through the application of CL. Through comparisons, we observe that on the Mutag dataset, which contains multi-attributed graphs, the improvement from incorporating line graphs is greater than the improvement from introducing CL. However, on the Regular dataset, which does not include multi-attributed graphs, introducing CL yields better results. This further illustrates the effectiveness of the line graph in leveraging its advantages on multi-attributed graphs, bringing abundant information into graph networks to achieve better results.

5 Conclusions

In this research, we propose a novel model for subgraph matching on multi-attributed graphs. We improve line graphs to acquire the features on nodes and edges, and construct a CL framework to enhance the quality of embeddings, thus improving the model's ability to predict subgraph matches. Experimental results confirm the feasibility and effectiveness of our approach.

References

[1] LAN Z X, YU L M, YUAN L L, et al. SubGMN: the subgraph matching network model [EB/OL]. (2019-03-15) [2024-09-10]. https://arxiv.org/abs/2104.00186v3.

[2] LAN Z X, MA Y, YU L M, et al. AEDNet: adaptive edge-deleting network for subgraph matching[J]. *Pattern Recognition*, 2023, 133: 109033.

[3] LIU X, SONG Y Q. Graph convolutional networks with dual message passing for subgraph isomorphism counting and matching [J]. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022, 36(7): 7594-7602.

[4] BARNES J A, HARARY F. Graph theory in network analysis[J]. *Social Networks*, 1983, 5(2): 235-244.

[5] XIAO S X, WANG S P, DAI Y F, et al. Graph neural networks in node classification: survey and evaluation[J]. *Machine Vision and Applications*, 2021, 33(1): 4.

[6] HANG M H, CHEN Y X. Link prediction based on graph neural networks[EB/OL]. (2018-02-27) [2024-09-10]. https://arxiv.org/abs/1802.09691v1.

[7] PRANNAY K, PIOTR T, CHEN W, et al. Supervised contrastive learning[J]. *Advances in Neural Information Processing Systems*. 2020, 33: 18661-18673.

[8] HE K M, FAN H Q, WU Y X, et al. Momentum contrast for unsupervised visual representation learning [C]//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). New York: IEEE, 2020: 9726-9735.

[9] HAFIDI H, GHOGHO M, CIBLAT P, et al. GraphCL: contrastive self-supervised learning of graph representations[EB/OL]. (2020-08-25) [2024-09-10]. https://arxiv.org/abs/2007.08025.

[10] QIU J Z, CHEN Q B, DONG Y X, et al. GCC: graph contrastive coding for graph neural network pre-training[C]//Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York: ACM,

2020: 1150-1160.
[11] ULLMANN J R. An algorithm for subgraph isomorphism[J]. *Journal of the ACM*, 1976, 23(1): 31-42.
[12] CORDELLA L P, FOGGIA P, SANSONE C, et al. A (sub) graph isomorphism algorithm for matching large graphs[J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004, 26(10): 1367-1372.
[13] ZOU L, CHEN L, YU J X, et al. A novel spectral coding in a large graph database[C]//Proceedings of the 11th International Conference on Extending Database Technology Advances in Database Technology-EDBT' 08. New York: ACM, 2008: 181-192.
[14] SHASHA D, WANG J T L, GIUGNO R. Algorithmics and applications of tree and graph searching[C]//Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. New York: ACM, 2002: 39-52.
[15] CHOI D, LEE H, LIM J, et al. Efficient continuous subgraph matching scheme considering data reuse[J]. *Knowledge-Based Systems*, 2023, 282: 111120.
[16] LIU L H, DU B X, XU J J, et al. G-finder: approximate attributed subgraph matching[C]// 2019 IEEE International Conference on Big Data. New York: IEEE, 2019: 513-522.
[17] BHATTARAI B, LIU H, HUANG H H. CECI: compact embedding cluster index for scalable subgraph matching[C]//Proceedings of the 2019 International Conference on Management of Data. New York: ACM, 2019: 1447-1462.
[18] ARAI J, FUJIWARA Y, ONIZUKA M. GuP: fast subgraph matching by guard-based pruning [J]. *Proceedings of the ACM on Management of Data*, 2023, 1(2): 1-26.
[19] MA Y X, XU B M, YIN H F. Tps: a new way to find good vertex-search order for exact subgraph matching[J]. *Multimedia Tools and Applications*, 2024, 83(27): 69875-69896.
[20] HE J Z, CHEN Y X, LIU Z Y, et al. Optimizing subgraph retrieval and matching with an efficient indexing scheme[J]. *Knowledge and Information Systems*, 2024, 66(11): 6815-6843.
[21] YANG F, ZHANG H Y, TAO S M. Semi-supervised classification via full-graph attention neural networks[J]. *Neurocomputing*, 2022, 476: 63-74.
[22] SOCHER R, CHEN D Q, MANNING C D, et al. Reasoning with neural tensor networks for knowledge base completion[J]. *Advances in Neural Information Processing Systems*, 2013, 12: 926-934.
[23] YING R, LOU Z Y, YOU J X, et al. Neural subgraph matching[EB/OL]. (2020-07-06)[2024-09-10]. https://arxiv.org/abs/2007.03092.
[24] CEN Y K, ZOU X, ZHANG J W, et al. Representation learning for attributed multiplex heterogeneous network[C]//Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York: ACM, 2019: 1358-1368.
[25] SCHLICHTKRULL M, KIPF T N, BLOEM P, et al. Modeling relational data with graph convolutional networks[M]//Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018: 593-607.
[26] VASHISHTH S, SANYAL S, NITIN V, et al. Composition-based multi-relational graph convolutional networks[EB/OL]. (2019-11-08)[2024-09-10]. https://arxiv.org/abs/1911.03082v2.
[27] CAO J H, FANG J Y, MENG Z Q, et al. Knowledge graph embedding: a survey from the perspective of representation spaces[J]. *ACM Computing Surveys*, 2024, 56(6): 1-42.
[28] CARLETTI V, FOGGIA P, SAGGESE A, et al. Introducing VF3: a new algorithm for subgraph isomorphism[C]//International Workshop on Graph-Based Representations in Pattern Recognition. Cham: Springer, 2017: 128-139.
[29] TAUD H, MAS J F. Multilayer perceptron (MLP)[M]// Geomatic Approaches for Modeling Land Change Scenarios. Cham: Springer, 2018: 451-455.
[30] OKU H, ISHIKAWA M. High-speed liquid lens with 2 ms response and 80.3 nm root-mean-square wavefront error[J]. *Applied Physics Letters*, 2009, 94(22): 221108.
[31] LOSHCHILOV I, HUTTER F. Decoupled weight decay regularization[EB/OL]. (2019-01-04)[2024-09-11]. https://doi.org/10.48550/arxiv.1711.05101.

基于对比学习的多属性图子图匹配

刘博智，方　秀，孙国豪*，陆金虎
东华大学 计算机科学与技术学院，上海 201620

摘　要：图被广泛应用于从化学信息学到社会网络分析等领域。与图相关的问题变得越来越重要，子图匹配是其中最具挑战性的任务之一。子图匹配的目的是在数据图中找到与查询图同构的所有子图。传统的搜索方法大多依赖于计算复杂度高的搜索策略，难以用于大规模的真实数据集。随着图神经网络（graph neural network，GNN）的出现，研究人员开始用其来解决子图匹配问题。然而，在图的学习过程中，忽略了节点和边上的属性特征，这导致在现实场景中匹配结果的不准确。为解决这个问题，我们提出了一种新模型，即多属性图子图匹配网络（subgraph matching on multi-attributed graph network，SGMAN）。SGMAN首先利用改进的线图来捕获节点和边上的特征，进而将 GNN 和对比学习（contrastive learning，CL）相结合，得到图的嵌入表示，并计算匹配矩阵以表示匹配结果。在公开数据集上的实验结果证实了该模型的优越性能。

关键词：子图匹配；图神经网络；多属性图；对比学习