Contrastive Graph Similarity Networks

LUZHI WANG[*], College of Intelligence and Computing, Tianjin University, China
YIZHEN ZHENG[*], Department of Data Science and AI, Faculty of IT, Monash University, Australia
DI JIN, College of Intelligence and Computing, Tianjin University, China
FUYI LI, College of Information Engineering, Northwest A&F University, China
YONGLIANG QIAO, Australian Centre for Field Robotics, The University of Sydney, Australia
SHIRUI PAN[†], School of Information and Communication Technology, Griffith University, Australia

Graph similarity learning is a significant and fundamental issue in the theory and analysis of graphs, which has been applied in a variety of fields, including object tracking, recommender systems, similarity search, etc. Recent methods for graph similarity learning that utilize deep learning typically share two deficiencies: (1) they leverage graph neural networks as backbones for learning graph representations but have not well captured the complex information inside data, and (2) they employ a cross-graph attention mechanism for graph similarity learning, which is computationally expensive. Taking these limitations into consideration, a method for graph similarity learning is devised in this study, namely, **C**ontrastive **G**raph **Sim**ilarity Network (CGSim). To enhance graph similarity learning, CGSim makes use of the complementary information of two input graphs and captures pairwise relations in a contrastive learning framework. By developing a dual contrastive learning module with a *node-graph matching* and a *graph-graph matching* mechanism, our method significantly reduces the quadratic time complexity for cross-graph interaction modeling to linear time complexity. Jointly learning in an end-to-end framework, the graph representation embedding module and the well-designed contrastive learning module can be beneficial to one another. A comprehensive series of experiments indicate that CGSim outperforms state-of-the-art baselines on six datasets and significantly reduces the computational cost, which demonstrates our CGSim model's superiority over other baselines.

CCS Concepts: • **Information systems → Data mining**.

Additional Key Words and Phrases: Graph Similarity Learning, Graph Neural Networks, Contrastive Learning

[*]Both authors contributed equally to this research.
[†]Corresponding author.

Authors' addresses: Luzhi Wang, College of Intelligence and Computing, Tianjin University, Tianjin, Tianjin, China, wangluzhi@tju.edu.cn; Yizhen Zheng, yizhen.zheng1@monash.edu, Department of Data Science and AI, Faculty of IT, Monash University, Australia; Di Jin, College of Intelligence and Computing, Tianjin University, Tianjin, Tianjin, China, jindi@tju.edu.cn; Fuyi Li, College of Information Engineering, Northwest A&F University, Yangling, Shaanxi, China, fuyi.li@nwafu.edu.cn; Yongliang Qiao, Australian Centre for Field Robotics, The University of Sydney, Sydney, NSW, Australia, yongliang.qiao@ieee.org; Shirui Pan, School of Information and Communication Technology, Griffith University, Australia, s.pan@griffith.edu.au.

1559-1131/2023/1-ART $15.00
https://doi.org/10.1145/3580511

1 INTRODUCTION

Learning a function to calculate how similar two graphs are to each other is considered as the main objective of graph similarity learning (GSL). It is a prominent graph theory issue with numerous practical applications, including computer vision [35], programming code analysis [47], financial transaction analysis [22], protein-protein interaction alignment [5], and entity linking in knowledge graphs [46]. Numerous graph-matching techniques have been proposed to evaluate how similar two graphs are to each other. Some traditional methods for graph similarity computation, such as heuristic search methods and exactly search strategies, focus only on the graph topology. For instance, using random walks as its backbone, a graph matching study for finding exact graph matching is introduced by the work of Gori et al. [10]. By separating colliding graph signatures, Gori et al. [11] provide a search algorithm to locate full mappings between nodes that need to be matched. These techniques are designed to identify the node-to-node (N2N) complex correspondence between two given graphs.

However, exact graph matching, finding exactly search results for node correspondence in the graph, commonly referred to as the graph isomorphism problem or the subgraph isomorphism problem (to identify if a graph to a part of another graph), is computationally expensive and even NP-complete [13]. To solve this challenge, relaxed graph matching, which approximately computes the similarities, has drawn much attention. Hlaoui et al. [16] propose a search method that decomposes the matching process into several stages to find the smallest error mapping for the relaxed graph matching problem. This algorithm significantly reduces the search space and produces good approximate matches between graphs. Zhou et al. [58] formalize pair-wise graph matching into a quadratic assignment problem (QAP) and design a fast approximation algorithm by decomposing the affinity matrix, a matrix for structuring the similarity, into Kronecker products of some special and small matrices. Lagrangian relaxation graph matching (LRGM), a revolutionary and cost effective graph matching relaxation technique, is proposed by Jiang et al. [17]. LRGM makes an effort to offer frameworks for maximizing the relaxation matching goal, by including affine mapping constraints into the matching target. The aforementioned works focus on finding node-to-node matches on the graph topology. However, some inherent property features (e.g. node/edge weights) contained in graphs are overlooked, which limits the comparison of graph similarity computing.

GNNs, or graph neural networks, have emerged recently as viable methods for GSL. GNN methods [41] typically employ a GNN encoder for that each node in a graph is generally encoded into a latent space with low-dimension by capturing the topological structure information and node content (if exists). Therefore, GNN-based graph similarity learning methods can evaluate graph similarity more comprehensively than methods that only consider graph topology information [2]. Graph-level representations are taken as foundations into consideration, in order to measure the similarity score between the given two graphs. There are many ways to obtain a graph-level representation, and most works use a pooling layer to obtain graph-level embeddings. The pooling layer obtains a graph-level representation which is a dimension-fixed single vector, by aggregating all node embeddings in the graph. The simplest pooling layer can be the mean pooling, which regards the graph-level representation as a fixed-dimension vector by averaging all node representations in the pooled graph. In addition to mean pooling, max pooling and LSTM pooling are the popular pooling methods for obtaining graph-level representations. However, learning the representation of every single graph is not sufficient to compare the difference well between two graphs. For this reason, some works refine the difference between two graphs by comparing nodes one by one in two input graphs. Specifically, to capture the correspondence between nodes across graphs, these methods employ a cross-graph attention mechanism for graph representation learning.

While achieving promising results, these methods typically suffer from two deficiencies. *Limitation 1:* the graph representation learning (GRL) module in these methods do not well capture the complex information for graph representation learning. For example, vanilla GNN encoders, such as GGNN [27], GCN [24], GAT [37], are designed for general graph representation learning and overlook pairwise relations between graphs. Moreover, the aforementioned works generally take a single graph as input, whereas GSL focuses on comparing a pair of graphs. Thus, instead of learning a pair of graphs separately, it is preferable to learn these graph representations simultaneously. *Limitation 2: computationally expensive.* Learning the matching relationship between nodes is a key part of improving similarity learning. To determine how a node corresponds to other nodes in another graph, one node in a graph has to be attended to over all nodes in another graph in the existing methods (as shown in Fig. 1 (a)), resulting in a time complexity of $O(|V_1||V_2|)$ for a single epoch in the cross-graph attention mechanism, where $|V_1|$ and $|V_2|$ are node numbers in two matched graphs [26]. It is challenging to apply this type of cross-graph matching mechanism to address real-world issues since it is computationally expensive.

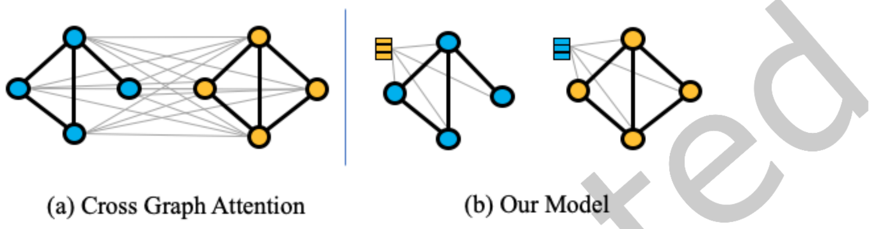


Fig. 1. Comparison of different cross-graph interaction mechanisms. (a) Existing methods employ a *1-versus-all* attention for every node to capture the correspondence between this node and all other nodes in another graph, resulting in a complexity of $O(|V_1||V_2|)$. (b) Our method employs a *1-versus-1* mechanism, where each node will be paired with the whole graph for contrastive learning, resulting in a complexity of $O(|V_1| + |V_2|)$. $|V_1|$ and $|V_2|$ are node numbers in two matched graphs.

To address the aforementioned limitations, we suggest a novel method, **C**ontrastive **G**raph **Sim**ilarity network (CGSim), which makes use of contrastive learning to both enhance graph representation learning and reduce the computational complexity in pairwise matching. Contrastive learning, as an unsupervised learning framework, has achieved promising performance in diverse tasks such as natural language processing (NLP) [9], graph analysis on representation learning for sparse graphs [21], and computer vision (CV) [54]. The primary objective underlying contrastive learning is to create a pair of semantically similar instances from the data itself, thus, the mutual information between the data and the constructed similar instances can be maximized. By using contrastive learning, we can easily extract the agreement between two instances, that is, the similarity between them. Contrastive learning has shown its effectiveness in exploiting the complex information in (graph) data, however, all existing studies, such as [14, 59], focus on building pairwise contrastive samples within a single graph. The cross-graph contrastiveness, which is the natural case for GSL, has not been exploited. Specifically, contrastiveness in existing studies is mostly built between a graph and its augmented graph, while in our study we build contrastiveness between two different graphs, which further takes the similarity between nodes into account for GSL.

As part of this work, we develop a new method for learning similarity between graphs based on cross-graph contrastive learning. To address *Limitation 1* of existing approaches, our method directly captures the pairwise relations between graphs by exploiting the complementary information from the different graphs. In particular, we compute the similarity of a pair-wise node from the two different graphs using contrastive learning. To reduce the computational cost in matching, i.e., *Limitation 2*, we design our method as a dual contrastive learning framework, *i.e.,* employing *node-graph matching* and *graph-graph matching* simultaneously. For node-graph matching, given two graphs G_1 and G_2, our method CGSim will create a node pair (v, G_2) for every node v in G_1 for contrastive learning. As CGSim replaces the computationally expensive *1-versus-all* attention mechanism with a *1-versus-1* contrastive learning module, it significantly reduces the time complexity for cross-graph interaction modeling from $O(|V_1||V_2|)$ to $O(|V_1| + |V_2|)$, as shown in Fig. 1 (b). To provide a measure of the similarity of two graphs globally, we provide graph-graph matching, which uses graph-level representations to compute similarity. More specifically, our method maximizes the mutual information of the two representations from two similar graphs, which further enhances the representation learning for two graphs. Our graph representation learning and contrastive learning are learned in a unified framework so that both GSL and contrastive learning can be beneficial and enhance one another. In particular, the similarities between the two graphs are extracted, as well as the differences between them are expanded. Employing six real-world datasets for evaluation, we show that CGSim achieves consistent performance gains compared with baselines. We also provide complete ablation experiments for demonstrating the validity of our proposed cross-graph comparability. We provide runtime tests to demonstrate that our model alleviates the limitation introduced in Section 1 and has a lower runtime, which makes the GSL algorithm more applicable to the real world. We summarize contributions of this paper as follows:

- To obtain a finely vectorized representation of nodes, we integrate contrastive learning into the representation learning process of GSL. CGSim employs a cross-graph learning scheme to contrastively enhance representation learning.
- To obtain a legible matching relationship, we propose a dual contrastive matching framework that employs *node-graph matching* and *graph-graph matching* simultaneously for GSL. CGSim significantly reduces the time complexity for cross-graph interaction modeling. Our proposed model differs significantly from existing approaches in this way.
- We evaluate our algorithm with extensive experiments and compare it with various baselines. The results demonstrate that superiority of CGSim.

2 RELATED WORK

2.1 Pairwise graph similarity computation

In graph theory, pairwise graph similarity calculation is a fundamental problem, which is employed to determine how similar a pair of graphs are. It benefits many real-world application tasks such as object tracking, recommender systems, binary code analysis, and similarity search. The establishment of similarity measures is adopted in early works. For example, Zager et al. [52] employ a linear update to obtain node similarity scores and edge similarity scores. Papadimitriou et al. [33] propose five similarity evaluation methods for web graph anomaly detection. These strategies have been widely adopted in the follow-up works [25]. However, the aforementioned techniques mainly rely on original graph architectures and they are learning-free, making it challenging to effectively utilize features provided by nodes and edges. Benefiting from the deep learning advancement, dramatic development has been made by graph similarity computation [40, 53]. Some GSL works rely on computer vision, and use encoders to extract graphs containing attributes from input images, then perform GSL, and design loss-optimized computer vision encoders. For example, Yu et al. [50] convert the Hungarian algorithm to a hard attention mechanism and incorporate a multi-headed attention mechanism for optimizing deep encoders in deep graph matching. An entire end-to-end pipeline for deep networks modeled by Wang et al. [39] is differentiable to parameterize the affinity functions within and across graphs to learn graph similarity. As the advancement of deep encoders inside the CV field, deep graph kernels have been the focus of some recent studies. Though learning a designed kernel function, deep graph kernels measure the similarity scores in substructure space among any two matched graphs [30].

Kernel-based techniques use kernel functions that correspond to the inner product of Reproducing Kernel Hilbert Space (RKHS) to analyze the similarity of two objects. The challenge for kernel-based techniques is to develop an appropriate kernel function that captures the information of the structures while facilitating computation. Yanardag et al. [48] propose a unified framework for learning potential representations of graph substructures, the DGK framework. The framework defines graph-to-graph similarity by using the information on dependencies between substructures to define the potential representation of graph structures by similarity. The Weisfeiler-Lehman(WL) subtree kernel is a commonly used graph kernel basis. Apart from that, the authors give special cases of other two common kernel application frameworks: Graphlet kernels and shortest path-based graph kernels. An unsupervised graph representation learning approach comes from Alrfou et al. [1], attempts to build a learnable encoder to capture the graph structure. A cross-graph attention network is trained for recording the interaction between two embeddings of each graph pair. They use the predictions of the attention-enhanced encoder to define the divergence scores of each pair of graphs. Finally, they use these pairwise divergence scores to construct an embedding space for all graphs, reducing the reliance on domain-specific knowledge (such as, on the WL kernel).

Different from deep graph kernel methods, GNN-based methods are usually based on graph representation learning. GNN-based methods usually use a GNN as the base encoder. Then a cross-graph attention mechanism is performed by GNN-based methods to learn the extent of the nodes matched with each other between two matching graphs [19]. Concretely, SimGNN [2], combines two strategies. Firstly, in order to offer global information about a graph, SimGNN constructs an embedding function for transforming the input graph pairs into an embedding space. Secondly, a pairwise node-level comparison framework is created by SimGNN to supplement graph embeddings with fine-grained node information. Finally, the GNN encoder is optimized through a loss carefully designed for the graph similarity problem. In GMN [26], they propose a GSL framework, which receives graph pairs as inputs. By using a matching mechanism based on a cross-graph attention strategy, the GMN framework then calculates the pairwise graph's similarity score. The matching mechanism then does joint inference on the pair. Afterward, considering that these graph neural networks are limited by the fixed dimension of graph representation, GraphSim [3] finds that the fixed-dimensional graph-level representations may not fully capture different graphs sizes and structures. Therefore, GraphSim designs a node multi-scale graph representation for similarity computation to find fine-grained differences between two graphs. GraphSim breaks the idea of fixed-dimensional vectors, and directly matches two groups of node embeddings to represent the entire graph. MGMN [28] finds that recent works on GSL mainly consider graph-to-graph (G2G) or node-to-node (N2N) interactions while ignoring the cross-level interactions (such as, node-to-graph (N2G) interaction information). With end-to-end computation of graph similarity between the two graphs, MGMN suggests a way for effectively learning cross-level graph interactions between nodes in the two different graphs. Unfortunately, the aforementioned models have a common limitation in cross-graph attention mechanism, which is computationally expensive.

2.2 Graph representation learning (GRL)

Recently, GRL focuses on learning graph representations by utilising both the topology and attributed information of graphs [18, 51]. It has been applied in many real-world scenarios, such as recommender systems (RSs), and social networks (SNs) [20]. In early works, DeepWalk [34] vectorizes the graph through a random walk algorithm with few network annotation nodes to learn social network embeddings. The key idea of it is inspired by the Skip-gram method of word2vec in NLP [32]. In order to handle large-scale graph data, LINE [36] maps all of the network's nodes into an embedding space and then attempts to preserve the network's original structure in terms of first-degree and second-degree links. The representation of the network nodes is then modeled by processing the word vectors. Graph neural networks (GNNs) have garnered a fair amount of attention as a result of the advancement of deep graph learning [43, 44, 55]. For instance, a technological architectural domain message-passing method is the gated graph neural network (GGNN), which takes the Gate Recurrent Unit (GRU) as the foundations. Three categories of processing make up the general methodology of message passing: message passing, update, and read operation. The node's embedding in the next moment is determined by its current moment embedding, as well as the node's neighbor's current moment embeddings and the edges information between the two nodes interacting. Graph Convolutional Network (GCN) [24] aggregates the node features with the central node's neighbors through a weighted average function to create a new representation of the node in the spectral domain. Subsequently, there are many variants methods of GCN have been studied, for example, graph attention network (GAT) [37], graph isomorphism Network (GIN) [45]. GAT provides a self-attentive mechanism for aggregating neighbor node information in order to improve the method's accuracy, resulting in the adaptive matching of weights to different neighbors. GAT can give various weights to different nodes in the neighborhood without any kind of expensive matrix operations (e.g., inversion) or depending on the pre-knowledge of the graph structure. GIN analyzes GNNs based on graph isomorphism theory, and proves that the Weisfeiler-Lehman (WL) test is the upper bound of GNN performance. Graph isomorphism is also an important indicator of GSL and the Weifeiler-Lehman test is a powerful algorithm for distinguishing graph structures and can discriminate whether a graph is isomorphic or not. GIN demonstrates that it has equivalent expressive/discriminative abilities to the WL test.

2.3 Contrastive learning

In self-supervised learning, contrastive learning is an emerging technology that cannot be ignored and originates from computer vision. Contrastive learning compares minute variations across samples in the embedding space for differentiating representation of samples. Recent years have seen a lot of fascinating works. For example, MOCO [15] translates contrastive learning into a dictionary lookup system, where they construct dynamic dictionaries with queues and average-shift encoders. This allows the construction of large and consistent dictionaries to facilitate contrastive unsupervised learning. SimCLR [6] simplifies recently proposed contrastive self-supervised learning algorithms without the need for specialized architectures or memory banks. SimCLR demonstrates that developing effective prediction tasks depends heavily on data augmentation and introduces learnable nonlinear transformations between representations and contrastive losses, which greatly contributes to the quality of learned representations. This is not only true for color distortion, but also for other types of data transformations. In general, contrastive training is more sensitive to systematic bias in the data. In machine learning, data bias is a widespread problem, which has a greater impact on contrastive methods (e.g. color distortion). Grill et al. [12] propose BYOL, a contrastive learning method that can not rely on negative sampling and thus avoids the data bias problem. BYOL does not care whether different samples have different representations (i.e., the contrast part of contrastive learning), but simply makes similar samples similarly represented. It may seem inconsequential, but such a setup will significantly improve model training efficiency and generalization. In training, each sample is only sampled once per traversal, and there is no need to focus on negative samples. Since no negative sampling is required, BYOL has higher training efficiency. The BYOL model is insensitive to the systematic bias of the training data, which means that BYOL can have better applicability to unseen samples as well. Barlow Twins [54] doesn't take into account many complex tricks, such as data augmentation, negative samples, networks of momentum update, predictor and stop gradient operations. Barlow Twins tries to learn the representation from a different perspective, starting from the embedding itself, rather than from the samples. The optimization goal is to make the correlation matrix of the features in different perspectives close to the constant matrix, i.e., making the features in different dimensions represent different information as much as possible can improve the representational power of the features. Given the excellent performance of contrastive learning, some work has introduced contrastive learning into the supervised models, and excellent results have been obtained. For example, to use label information efficiently, some researchers, i.e., Khosla et al. [23], recently adapts a self-supervised contrastive learning method for properly using in a fully supervised setting. In particular, nodes with the same label are brought together while clusters of data from distinct categories are pushed apart in the embedding space.

In addition to computer vision, contrastive learning is an emerging approach for GRL and it works for self-supervised learning [56, 57]. In contrastive learning, the principal objective is to continue improving the mutual information between the two semantically similar instances that have been constructed from the data itself. Contrastive learning reaches state-of-the-art performance [6, 29, 42, 56] and has been applied to graph data. For instance, DGI [38] maximizes the mutual information of local information and global information to enhance representations. By discriminating between the local-global representations, MVGRL [14] is presented in a new way by using contrastive multi-view representation learning for the graph representation learning underlying the DGI idea. GraphCL [49] proposes a graph-level contrastive learning method with four types of graph augmentation technologies. However, all contrastive-learning-based methods are only designed for a single graph representation. This scheme is not suitable for GSL, which normally takes two graphs as input to compare the difference. To fill this gap, we propose CGSim, which can build contrastiveness between different graphs.

Notations	Descriptions
G	A graph.
V, E, X	Node sets, edge sets and feature sets.
h_v	Embedding of a node v.
h_G	Embedding of a graph G.
l	l-th layer of neural network.
τ	Temperature parameter.
$\|V\|$	The cardinality of a node set V.
$N(v)$	The neighbor of a node v.
$AGG(\cdot)$	The aggregate function.
$Trans(\cdot)$	The transform function.
$AVG(\cdot)$	The average function.
$\theta(\cdot)$	Cosine similarity function.
$\mathbb{E}(\cdot)$	Exponential function.
$BCE(\cdot)$	Binary cross entropy loss.

Table 1. Summary of notations.

3 METHOD

Beginning with a definition of the graph similarity problem, this section introduces our proposed method CGSim, followed by the elaboration of each critical component. We list the key terms and notations in Table 1.

3.1 Problem Definition

Suppose that $G = (V, E, X)$ be an undirected graph with a node set V and an edge set E, where $E \subseteq V \times V$. $|\cdot|$ is denoted as the cardinality of numbers. Specifically, $|V|$ represents the number of nodes. $X \in \mathbb{R}^{|V| \times d}$ denotes features of nodes, where d is features' dimension. The similarity scores between two input graphs are defined in terms of the following:

Definition 3.1. **Graph similarity score.** Given two graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ as inputs, and their embeddings h_{G_1} and h_{G_2} are used to calculate the cosine similarity score $\theta(h_{G_1}, h_{G_2})$ between two graphs:

$$\theta(h_{G_1}, h_{G_2}) = \frac{h_{G_1} \cdot h_{G_2}}{\| h_{G_1} \| \cdot \| h_{G_2} \|}. \tag{1}$$

Our object is to train a learnable GNN encoder that regards a pair of graphs as input, and outputs their low-dimensional representations for similarity computation. The trained encoder can then be applied to any unseen graph pairs for similarity calculation during the inference phase. The calculated two graph cosine similarity scores are used in subsequent downstream tasks related to graph similarity.

3.2 Solution Overview

According to the two limitations we mentioned in Section 1, we design the CGSim foundation upon the following intuitions:

Intuition 1. Responsibility of Representations. The graph similarity computation is conducted based on the graph embeddings of input pairs. To create unified dimensional representations that may be used to quantify similarity, graph pairs need to be first mapped to the same low-dimensional space to get unified dimensional representations. Thus, learning representations that well persist in the original graph's complexity information is essential to the success of this measurement. Graph-level representations are pooled from node-level representations. Therefore, using GNN to learn complex node-level representations to fully express the properties of nodes is one of the key designs in our similarity learning.

Intuition 2. Capability of Cross-graph Interaction. One of the important methods for improving the computation of graph similarity scores is to establish the correspondence between the nodes in two graphs. To enable effective graph matching, a cross-graph interaction module is required to exploit the pairwise relations for graph representation learning. As far as we know, current works normally employ attention mechanisms for cross-graph interactions between nodes. Specifically, each node in one graph needs to perform attention calculations with all nodes in the other graph. The time complexity of these cross-graph attention mechanisms is $O(|V_1||V_2|)$, which is not conducive to application in real-world datasets. This is because learning the *1-versus-all* correspondence (i.e., attending all nodes in another graph with a target node to discover the correspondence) is computationally expensive. The ideal cross-graph interaction module should have a low-computational cost as well as a high representation capability. Contrastive learning, which naturally takes two graphs as input, is a promising solution for cross-graph interaction.

Overall Framework. Following the intuitions, we combine graph representation learning with contrastive learning in a unified framework to learn graph similarity. The overview of our proposed model CGSim, which consists of four components: (a) input graphs, (b) GNN module, (c) cross-graph contrastive learning module, (d) combinatorial loss module, is shown in Fig. 2. To train our model, we first input the graph pairs into a GNN module, which outputs node-level embeddings. By aggregating node embeddings, a graph pooling technique can generate embeddings at the graph level. Then, we construct a *dual* contrastive learning framework with a *node-graph matching* and a *graph-graph matching* scheme utilizing the generated different level embeddings. Besides, we refine the generated embeddings by further exploiting the supervision signals with the BCE loss. We jointly train a combinational loss that consists of the aforementioned parts (i.e., node-graph contrastiveness, graph-graph contrastiveness, and cross-entropy loss), so that each component can be beneficial to another. We will discuss more detailedly about these components of CGSim in the following sections.

3.3 Representation Learning

As shown in Fig. 2 (a)(b), the first two components of our suggested method (input graphs and GNN module) are illustrated in this section. A siamese network is designed for the purpose of distinguishing visual differences in computer vision by measuring the similarity between two input images [4]. Two inputs are supplied into two neural networks of a siamese neural network. The inputs are mapped to the new space by the two neural networks with the same weight. The similarity of the two inputs is measured by the loss calculation. Generally, a siamese network includes two neural networks with the same parameters, which leads to more robust semantic similarity learning. The architecture of the siamese network natural fits for GSL, which can introduce inductive

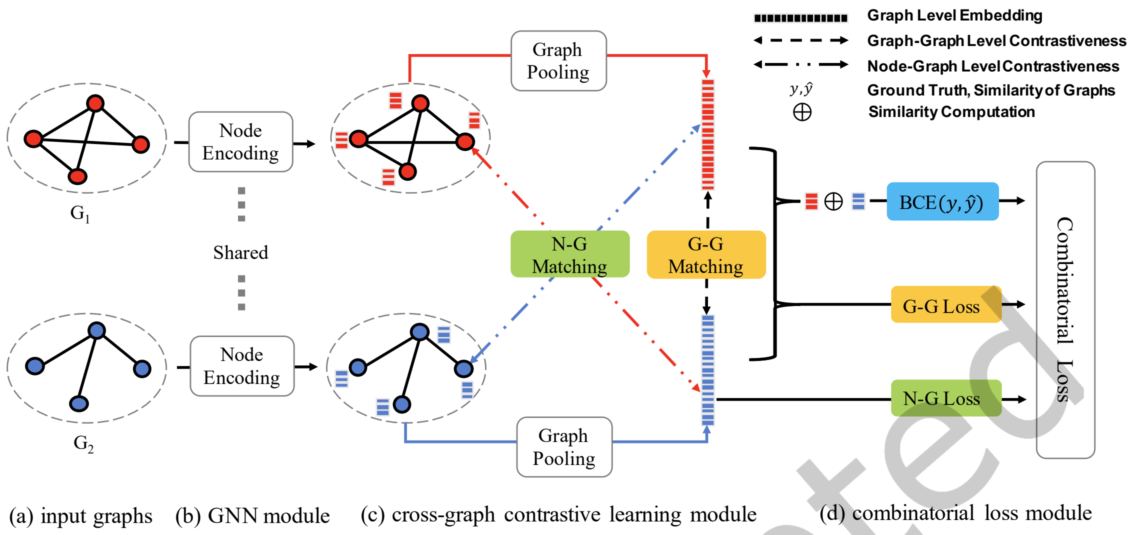

Graph Level Embedding
Graph-Graph Level Contrastiveness
Node-Graph Level Contrastiveness
$y, \hat{y}$ Ground Truth, Similarity of Graphs
$\oplus$ Similarity Computation
G1
G2
Node Encoding
Shared
Node Encoding
Graph Pooling
Graph Pooling
N-G Matching
G-G Matching
BCE$(y, \hat{y})$
G-G Loss
N-G Loss
Combinatorial Loss
(a) input graphs
(b) GNN module
(c) cross-graph contrastive learning module
(d) combinatorial loss module

Fig. 2. Overview of CGSim. The input graph pairs in (a) are sent to the GNN module (b) to capture the node-level representations by a shared encoder. The node-level representations are dubbed the input of module (c). In (c), graph pooling can be used to obtain graph-level representations. Then, both node-level representations, as well as graph-level representations, will be sent to a carefully designed cross-graph contrastive learning module to achieve cross-graph interaction by node-graph matching and graph-graph matching mechanisms. To refine the generated representations, supervision signals are further exploited with a cross-entropy loss. By penalizing the discrepancies between the predicted and ground-truth graph similarity scores, the BCE loss is calculated. The whole model is trained using the combinatorial loss module (d), which constitutes the node-graph matching loss, graph-graph matching loss, and cross-entropy loss collaboratively.

biases for identifying invariant patterns of similar objects [7, 8, 31]. In our method, we adopt a siamese network architecture as our backbone. Specifically, in our method, the GNN module is made up of two GNN encoders with shared learnable weights. To estimate the matching score of two graphs, we first need to embed two graphs into the vector space. Here, we employ a GNN encoder to acquire node representations by collecting nodes' local structure information. The GNN encoder may be considered a feature extractor for graphs, which mainly consists of message aggregation and transformation. Specifically, GNN updates node representations iteratively by aggregating its neighboring representations. The whole process is defined as:

$$\begin{aligned} h_v^{(l)} &= \mathtt{Agg}^{(l)}(h_u^{(l-1)} : u \in N(v)), \\ h_v^{(l)} &= \mathtt{Trans}^{(l)}(h_v^{(l-1)}, h_v^{(l)}), \end{aligned} \tag{2}$$

where $h_v^{(l)}$ is the representation vector of a node v at the l-th layer and $N(v)$ represents a node sets adjacent to v [45]. $\mathtt{Agg}(\cdot)$ is an aggregate function that aggregates the neighbors' information to the node itself and $\mathtt{Trans}(\cdot)$ is the transform function that transforms the message to update a new embedding. Then, the graph-level representation h_G can be obtained via mean-pooling, denoted as $\mathtt{Avg}(\cdot)$ below:

$$h_G = \mathtt{Avg}(h_v : v \in V). \tag{3}$$

The graph representation learning module is the basic framework of subsequent work. After obtaining graph-level and node-level representations, we perform subsequent cross-graph interaction computation in the following.

3.4 Cross-graph Contrastive Matching

One of the primary limitations of the cross-graph matching mechanism is high computational complexity $O(|V_1||V_2|)$, which limits the computing capacity of GSL. To confront this challenge, we introduce a low-complexity dual cross-graph contrastive learning module, which allows our model to learn cross-level interactions (i.e., node-graph matching and graph-graph matching) both effectively and efficiently. Graph-graph contrastiveness learns coarse-grained matching between graphs, whereas node-graph contrastiveness learns fine-grained matching between nodes and graphs. Our method reduces the complexity to $O(|V_1| + |V_2|)$, while maintaining competitive performance.

Contrastive Node-Graph Matching. To reduce the excessive calculational cost, we construct a contrastive node-graph matching mechanism for cross-graph information distillation instead of associating node representations in one graph to node representations in its counterpart pairwisely. Aggregating node embeddings in a graph yields a graph-level embedding, which is an average representation of the node embeddings, and the dimensions of it are the same to the node embeddings, we try to use a graph embedding of one graph to take the place of node embeddings in the other graph. To further explore the matching relations between node representations and graph representations, we construct a cross-graph contrastive learning method based on the following intuitions: if two graphs are similar, all node embeddings in G_1 should be close to the graph-level embedding of G_2. Conversely, if two graphs are different, node embeddings in G_1 and the graph embedding in G_2 should be pushed away from each other. To achieve such strategies, we exploit mutual information to investigate the correlation between the node embeddings h_v and the graph embedding h_G [6, 59]. For an input graph pair (G_1, G_2), we define the node-graph mutual information $\texttt{MI}_{NG}$ between two graphs as:

$$\texttt{MI}_{NG}(h_v, h_{G_2}) := \sum\nolimits_{v \in |V_1|} \mathbb{E}(\theta\left(h_v, h_{G_2}\right)/\tau), \tag{4}$$

where $\mathbb{E}(\cdot)$ is an exponential function and τ is a temperature parameter that regulates the model's sensitivity to negative samples. The temperature parameter's role is to modify the level of focus on challenging samples. The sample is distinguished from the most comparable other samples with greater care as the temperature parameter decreases. To guide message interaction, as shown in Fig. 3, we use supervision signals to distinguish whether the input pairs are positive (similar) or negative (dissimilar). In the training dataset, each pair of graphs contains a ground truth label, which represents whether the pair is similar or dissimilar. According to these ground truth labels, we regard similar graph pairs as positive samples, and dissimilar pairs as negative samples. A batch with size K contains positive graph pairs and negative graph pairs. We regard (h_{G_1}, h_{G_2}) as a graph pair, $(h_{G_1}, h_{G_2})_p$ as a positive pair and $(h_{G_1}, h_{G_2})_k$ as a graph pair in the batch. We expect to maximize the mutual information between node embeddings h_v in G_1 and the graph embedding h_{G_2} for positive pairs. For negative pairs, we should minimize the mutual information between h_v in G_1 and h_{G_2} in vector space. Therefore, we define the node-graph loss as:

$$\ell_{NG}(h_{G_1}, h_{G_2}) = -\log \frac{\sum_{p=1}^{P} \texttt{MI}_{NG}(h_v, h_{G_2})_p}{\sum_{k=1}^{K} \texttt{MI}_{NG}(h_v, h_{G_2})_k}, v \in G_1, \tag{5}$$

where P is the number of positive pairs in batch size K. By means of the negative, the node-graph loss maximizes the mutual information between positive sample pairs as well as minimizes mutual information between negative sample pairs. Noticed that operations on graph pairs should be symmetric for effects on balance. We defined the final loss ℓ_{NG} as the average of the positive pair to reduce inaccuracy:

$$\mathcal{L}_{NG}(h_{G_1}, h_{G_2}) = \frac{1}{2}\left[\ell_{NG}(h_{G_1}, h_{G_2}) + \ell_{NG}(h_{G_2}, h_{G_1})\right]. \tag{6}$$

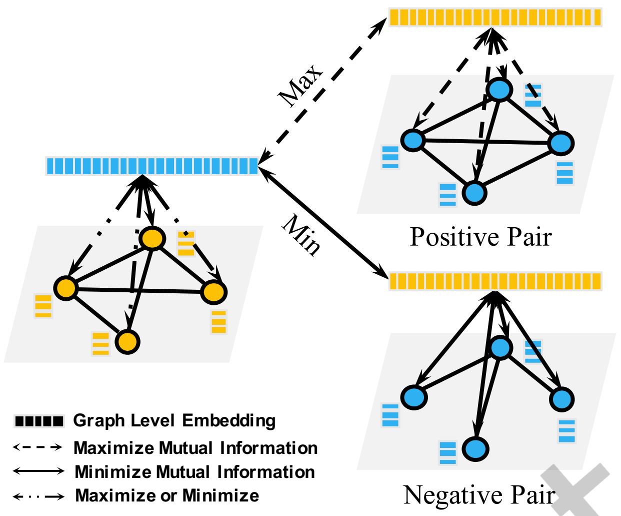


Fig. 3. A brief description of cross-graph contrastive learning module. The dotted line represents the cross-graph interaction.

Contrastive Graph-Graph Matching. We use node-level embeddings to learn cross-graph interactions with fine-grained contrastiveness, and the global information of graphs is also valuable that cannot be ignored. Thus, we further enrich the contrastiveness with a proposed graph-graph matching (i.e., contrasting between graph-level embeddings) to consolidate the learned representations of graphs. Intuitively, if two graphs are similar, they should keep a high agreement, that is, their mutual information should be maximized and vice versa. In one batch, for positive samples, we enlarge the agreement between the two graphs. For negative samples, we expand their differences macroscopically. We define the graph-graph mutual information $\mathtt{MI}_{GG}$ between two graphs as:

$$\mathtt{MI}_{GG}(h_{G_1}, h_{G_2}) := \mathbb{E}(\theta\left(h_{G_1}, h_{G_2}\right)/\tau). \tag{7}$$

Thus, the aforementioned processes can be formulated with the following loss functions:

$$\ell_{GG}(h_{G_1}, h_{G_2}) = -\log \frac{\sum_{p=1}^{P} \mathtt{MI}_{GG}(h_{G_1}, h_{G_2})_p}{\sum_{k=1}^{K} \mathtt{MI}_{GG}(h_{G_1}, h_{G_2})_k}. \tag{8}$$

Because the operations on G_1 and G_2 is interchangeable, we design the average loss to reduce the error, and the final objection is defined as:

$$\mathcal{L}_{GG}(h_{G_1}, h_{G_2}) = \frac{1}{2}\left[\ell_{GG}(h_{G_1}, h_{G_2}) + \ell_{GG}(h_{G_2}, h_{G_1})\right]. \tag{9}$$

To take full advantage of supervision signals, we formulate the optimization object by establishing the relationship between ground truth and the calculated similarity score. Specifically, we employ a binary cross-entropy loss (BCE Loss) to minimize distance among the calculated similarity score $\theta(h_{G_1}, h_{G_2})$ and the given similarity indicator y:

$$\mathcal{L}_{BCE}(h_{G_1}, h_{G_2}, y) = \sum_{k=1}^{K} \mathtt{BCE}(\theta(h_{G_1}, h_{G_2})_k, y). \tag{10}$$

The BCE loss provides a directed optimization goal for the guarantee of representation. It is discovered that the graph-graph matching loss differs from the BCE loss. Because the similarity score is used as the optimization goal in the BCE loss for graph representation learning. Graph-graph loss is a subsidiary part of the cross-graph matching module, which tries to learn complementary information from different graphs.

3.5 Objective Function

We design a combinatorial loss which is composed of (i) the node-graph contrastiveness loss (Equation 6), (ii) the graph-graph contrastiveness loss (Equation 9), and the BCE loss for similarity score adjustment (Equation 10) to train the whole model in an end-to-end manner. The goal is to minimize the combinatorial loss $\mathcal{L}$, which is calculated as follows:

$$\mathcal{L} = \alpha\mathcal{L}_{NG} + \beta\mathcal{L}_{GG} + \lambda\mathcal{L}_{BCE}, \tag{11}$$

where α, β and λ are weight parameters. We define that α, β and λ as tunable parameters to regulate the three losses' impacts ($\mathcal{L}_{NG}$, $\mathcal{L}_{GG}$, and $\mathcal{L}_{BCE}$). We set the constraint $\alpha+\beta+\lambda = 1$ to ensure these parameters are in range $[0, 1]$ and weightily summed to 1. This demonstrates how each loss function contributes to the overall loss. We employ the gradient descent to minimize $\mathcal{L}$ so that three kinds of losses can be trained simultaneously. Therefore, the graph representation learning part and the cross-graph contrastive learning part are not isolated but benefit from each other. The BCE loss optimizes the inputs of cross-graph contrastive learning, in turn, cross-graph contrastive learning also promotes the expression of nodes and graphs. Algorithm 1 depicts a brief description of CGSim.

Algorithm 1: The Contrastive Graph Similarity Network.

```
  Input: K graph pairs, temperature parameter $\tau$, the given similarity indicator $y$.
  Output: similarity score between graph pairs.
1 Calculate source node embeddings $h_v$ by using Eq. (2);
2 Calculate graph-level embeddings $h_G$ by using Eq. (3);
3 Calculate the node-graph mutual infomation $\mathtt{MI}_{NG}$ by using Eq. (4);
4 Calculate the node-graph matching loss $\mathcal{L}_{NG}$ by using Eq. (5, 6);
5 Calculate the graph-graph mutual infomation $\mathtt{MI}_{GG}$ by using Eq. (7);
6 Calculate the contrastive graph matching loss $\mathcal{L}_{GG}$ by using Eq. (8, 9);
7 Calculate the similarity indicator loss $\mathcal{L}_{BCE}$ by using Eq. (10);
8 Update model parameters using an optimization approach based on the loss defined in Eq. (11);
9 return the similarity scores between graph pairs;
```

Model advantages. We present the advantages between our proposed method CGSim and recent studies from two perspectives: (1) responsibility of representations and (2) capability of cross-graph matching. Firstly, CGSim integrates graph representation learning and cross-graph contrastive learning into a unified framework, which facilitates the message propagation in and cross graphs. Secondly, CGSim reduces the time complexity for cross-graph interaction modeling from $O(|V_1||V_2|)$ (*1-versus-all*) to $O(|V_1| + |V_2|)$ (*1-versus-1*). This is a significant improvement over existing methods such as GMN [26]. The experiments later will show the superior performance of our proposed framework.

4 EXPERIMENTS

Our goal in this section is to illustrate how effective CGSim works and how efficient it is by performing extensive experiments. We will answer the following critical research questions:

Q1: How effective is CGSim compared with the state-of-the-art (SOTA) GNN-based graphic similarity models?

Datasets	Graphs	AvgN	AvgE	Classes
OpenSSL [3, 200]	73,953	15.73	21.97	4,249
OpenSSL [20, 200]	15,800	44.89	67.15	1,073
OpenSSL [50, 200]	4,308	83.68	127.75	338
FFmpeg [3, 200]	83,008	18.83	27.02	10,376
FFmpeg [20, 200]	31,696	51.02	75.88	7,668
FFmpeg [50, 200]	10,824	90.93	136.83	3,178

Table 2. Datasets Description.

Q2: How do the proposed node-graph loss, graph-graph loss, and BCE loss help with the CGSim model?
Q3: What is the efficiency of the cross-graph contrastive learning module compared with that of existing cross-graph attention mechanisms?

Datasets Description. We implement GSL experiments on the binary code similarity detection datasets. As mentioned in the Introduction, binary code similarity detection is an essential application of GSL. The goal of binary code similarity detection is to check whether the control flow graphs (CFGs) of the two given binary functions are similar. Any pair of graphs contains a label y, implying that the pair of graphs are similar or dissimilar. For a given CFG pair (G_1, G_2), the similarity label $y = 1$ indicates G_1 and G_2 are similar, otherwise, $y = -1$ represents two CFGs are dissimilar [47]. We use the similarity score calculated by CGSim to predict whether the two graphs are similar.

We apply CGSim on datasets OpenSSL and FFmpeg [28]. Both OpenSSL and FFmpeg datasets have three subsets named with the graph size range. For example, in OpenSSL [3, 200], '3' means the minimum CFG size, and '200' means the maximum CFG size. Table 2 summarizes the statistics for datasets, where 'Graphs' donates the number of graphs in datasets, 'AvgN/AvgE' means the average of nodes/edges number in one dataset, and 'Classes' refers to the number of classes. It should be emphasized that the general datasets of graph classification tasks cannot be used for the graph similarity learning task. This is because the graph classification task only provides a label to each graph, whereas binary code similarity detection assigns a binary similarity label for two graphs instead of assigning two labels for two graphs.

Baselines. To verify the two intuitions we mentioned in Section 3.2, (1) responsibility of representation and (2) capability of cross-graph interaction, we evaluate CGSim from two perspectives, including graph representation learning and GSL. We compare CGSim with the SOTA graph representation learning methods: GGNN[27], GCN[24], GIN[45], and graph similarity learning methods, including SimGNN [2], GMN [26], GraphSim [3], and MGMN [28] as our baselines. MGMN contains three different versions, including MGMN (Max + BiLSTM), MGMN (FCMax + BiLSTM) and MGMN (BiLSTM + BiLSTM), where the contents in parentheses are combinations of different pooling operations. Here are the details of baseline algorithms and settings.

- **GGNN.** Gated graph neural network (GGNN) is a GRU-based classic spatial domain message passing model. We set the number of GGNN layers = 3, the hidden feature size = 100, learning rate = 0.0005, batchsize = 10.
- **GCN.** Graph convolutional network captures nodes' local structure in the spectral domain to learn high-level node representations. We employ a 3-layer GCN, and set the hidden feature size = 100, learning rate = 0.0005.
- **GIN.** In addition to distinguishing different graph structures, graph isomorphism networks that map similar structures to similar embeddings are able to capture their dependencies. We employ a 3 layers GIN with a hidden feature size = 100, learning rate = 0.0005.

- **SimGNN.** SimGNN designs a pairwise node embedding comparison matrix, and uses node level embedding to supplement the graph level representation by attention mechanism. In the experiment, we employ a 3-layer GCN as an encoder, the hidden feature size = 64, learning rate = 0.001.
- **GMN.** GMN designs a cross graph attention achieve the cross graph interaction. In the experiment, GMN applies a 1-layer multi-layer perceptron (MLP) as an encoder. The dimension of the node embedding is 32, learning rate = 0.001.
- **GraphSim.** GraphSim allows the comparison of graph similarity on multi-scales by using fixed dimension vectors. The learning rate = 0.001.
- **MGMN.** MGMN designs a multi-level graph matching network. MGMN provides three different versions with different aggregators, such as Max, FCMax, BiLSTM. To accommodate our experiments, we set the layer number of GNN encoder = 3, the hidden feature size = 100, learning rate = 0.0005.

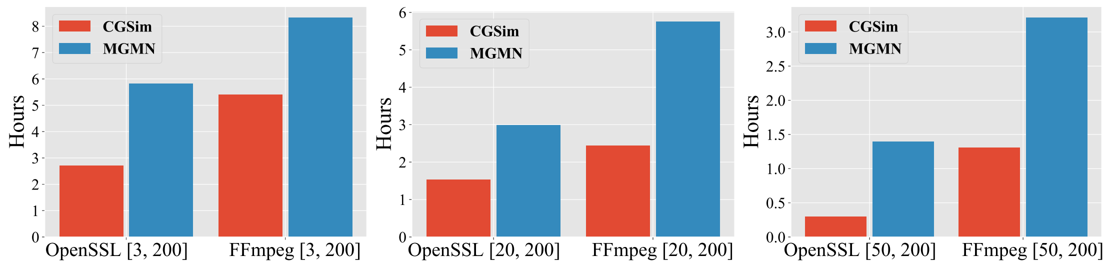
Model	**[3, 200]**	**OP**	**[20, 200]**	**OP**	**[50, 200]**	**OP**
GGNN	87.81±0.99	+11.1%	81.14±3.64	+20.4%	85.26±4.30	+14.6%
GCN	89.61±1.04	+8.1%	75.93±0.90	+28.6%	73.71±1.03	+32.5%
GIN	91.45±0.16	+6.6%	81.18±2.10	+20.3%	68.22±4.36	+30.1%
SimGNN	95.96±0.31	+1.6%	93.58±0.82	+4.4%	94.25±0.85	+3.7%
GMN	96.43±0.61	+1.1%	93.03±3.81	+5.0%	93.91±1.65	+4.0%
GraphSim	96.84±0.54	+0.7%	94.97±0.98	+2.9%	93.66±1.84	+4.3%
MGMN (Max + BiLSTM)	94.77±1.80	+2.9%	97.44±0.26	+0.2%	94.06±1.60	+3.9%
MGMN (FCMax + BiLSTM)	96.87±0.24	+0.6%	97.59±0.24	+0.1%	95.58±1.13	+2.2%
MGMN (BiLSTM + BiLSTM)	96.90±0.10	+0.6%	97.31±1.07	+0.4%	95.87±0.88	+1.9%
CGSim (Ours)	**97.51±0.44**	-	**97.68±1.32**	-	**97.70±1.28**	-

Table 3. The experimental results are presented as AUC scores on the OpenSSL datasets, which are expressed as a percentage (%).

Model	**[3, 200]**	**OP**	**[20, 200]**	**OP**	**[50, 200]**	**OP**
GGNN	91.22±0.01	+7.7%	81.14±3.64	+17.5%	87.29±1.11	+12.6%
GCN	94.65±0.01	+3.8%	92.73±0.85	+6.1%	89.06±2.56	+10.4%
GIN	95.08±0.50	+3.3%	92.52±0.51	+6.0%	84.96±0.78	+15.7%
SimGNN	95.38±0.76	+3.0%	94.31±1.01	+4.3%	93.45±0.54	+5.2%
GMN	94.15±0.62	+3.9%	95.92±1.38	+2.6%	94.76±0.45	+3.7%
GraphSim	97.46±0.30	+0.6%	96.49±0.28	+2.0%	94.48±0.73	+4.0%
MGMN (Max + BiLSTM)	97.44±0.32	+0.8%	97.84±0.40	+0.6%	97.22±0.36	+1.1%
MGMN (FCMax + BiLSTM)	98.07±0.06	+0.1%	98.29±0.10	+0.1%	97.83±0.11	+0.5%
MGMN (BiLSTM + BiLSTM)	97.56±0.38	+0.7%	98.12±0.04	+0.3%	97.16±0.53	+1.2%
CGSim (Ours)	**98.21 ±1.06**	-	**98.38 ± 0.91**	-	**98.30±0.70**	-

Table 4. The experimental results are presented as AUC scores on the FFmpeg datasets, which are expressed as a percentage (%).

Evaluation Metrics and Parameter Setting. We split each dataset into training, validation, and testing sets by 8 : 1 : 1, which follows the same data splits as in MGMN. We employ a 3-layer GGNN [27] as our graph



(a) Running time on [3, 200]. (b) Running time on [20, 200]. (c) Running time on [50, 200].

Fig. 4. Running time comparisons on different datasets.

encoder with the latent dimension setting as 100 and batch size as 16. We tune temperature parameter τ between 0 to 1 and batch size within $\{2, 4, 8, 16, 32\}$. The area under the ROC curve (AUC) is used as a metric of the performance, while the **bold** values indicate the best performance of approaches. The average AUC scores and standard deviations for each experiment are presented after five iterations of each experiment.

4.1 Q1: Effectiveness of CGSim

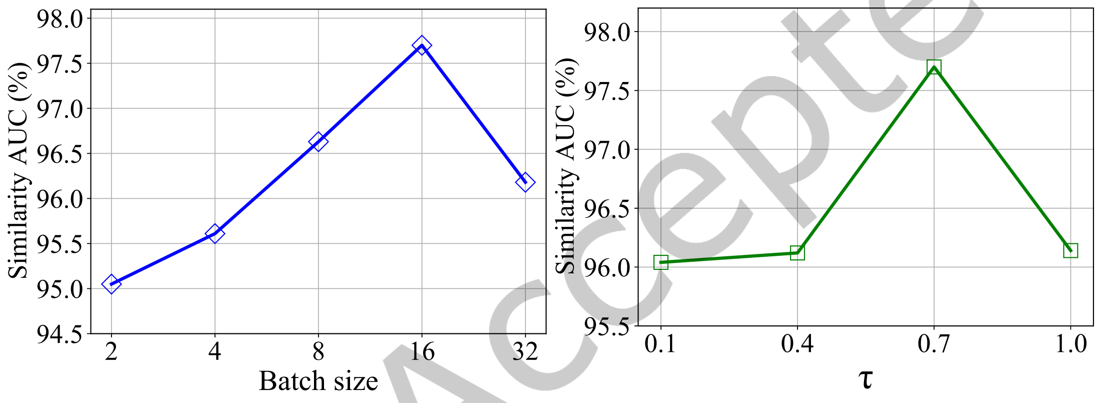
To respond to Q1, we evaluate CGSim comparing with the aforementioned baselines on various datasets. The experimental results for OpenSSL and FFmpeg are shown in Table 3 and Table 4. Column '[3, 200]' represents the AUC score on dataset [3, 200], '−' denotes no gain and '±' indicates a numerical range. Column 'Outperform (OP)' shows effectiveness of CGSim over other algorithms. Compared results are quoted from the MGMN [28]. The CGSim model performs the best compared with baselines using the AUC score. We can find those graph similarity methods, SimGNN, GMN, GraphSim and MGMN (Max + BiLSTM) performed comparatively poorly. CGSim outperforms all baseline methods, which can be attributed to the dual cross-graph contrastive mechanism. We develop a node-graph contrastive mechanism and graph-graph contrastive mechanism for cross-graph interaction, which not only promotes the message cross-graph propagation but also enriches the graph representation learning in the process of optimization. Though MGMN also considers a multi-level attention mechanism for cross-graph interaction, it heavily relies on the combination of different pooling operations, (i.e., BiLSTM with learnable parameters). With a naive pooling mechanism, mean pooling, CGSim achieves similar or better results than MGMN, which demonstrates the effectiveness of our proposed dual cross-graph contrastive learning. It is of importance to mention that our proposed method CGSim significantly improves graph similarity learning in OpenSSL [50, 200] and FFmpeg [50, 200], which indicates our approach works well on large graph-size datasets.

4.2 Q2: Contribution of $\mathcal{L}_{NG}$, $\mathcal{L}_{GG}$, $\mathcal{L}_{BCE}$

To analyze the contribution of $\mathcal{L}_{NG}$, $\mathcal{L}_{GG}$, $\mathcal{L}_{BCE}$, we conduct a combinational experiment on the parameters α, β and λ. We implement the CGSim on OpenSSL [50, 200]. Table 5 shows the average performance of five experimental iterations. For a single loss comparison, $\mathcal{L}_{GG}$ provides the best performance, which indicates graph-graph matching strategy $\mathcal{L}_{GG}$ is superior to the graph representation learning model without cross-graph interaction (i.e., $\mathcal{L}_{BCE}$). For a combination of two losses, the $\mathcal{L}_{NG} + \mathcal{L}_{BCE}$ achieve the highest performance. The best result is produced by the simultaneous application of three losses, which indicates that all mechanisms are effective. As long as $\alpha = 0.4$, $\beta = 0.4$ and $\lambda = 0.2$, CGSim achieves the best results 97.70 on OpenSSL [50, 200]. Accordingly, results can be concluded that both the dual cross-graph contrastive learning as well as graph representation learning can be used to improve the quality of graph similarity learning.

Loss	α	β	λ	AUC
$\mathcal{L}_{NG}$	1	0	0	91.22
$\mathcal{L}_{GG}$	0	1	0	94.05
$\mathcal{L}_{BCE}$	0	0	1	94.32
$\mathcal{L}_{NG} + \mathcal{L}_{GG}$	1/2	1/2	0	94.34
$\mathcal{L}_{NG} + \mathcal{L}_{BCE}$	1/2	0	1/2	96.88
$\mathcal{L}_{GG} + \mathcal{L}_{BCE}$	0	1/2	1/2	95.43
$\mathcal{L}_{NG} + \mathcal{L}_{GG} + \mathcal{L}_{BCE}$	1/3	1/3	1/3	**97.21**

Table 5. Effectiveness of losses on OpenSSL [50, 200].



(a) Effect on batch size. (b) Influence on τ.

Fig. 5. Influence on batch size and τ.

4.3 Q3: Efficiency of CGSim

In this section, we choose MGMN as the baseline to be compared with CGSim. This is because MGMN is the best-performed baseline according to Table 3 and 4. Furthermore, MGMN is one of the GSL baselines with the lowest time complexity [28]. We record the running time of training 100 epochs on the cross-graph interaction procedure. Fig. 4 shows that CGSim consumes much less time in cross-graph interaction. Compared with MGMN, the speed of CGSim is improved by 2 to 4 times in general, which is a huge improvement. As a result of the experiment, we also demonstrate that CGSim has better efficiency than the baselines we comparing.

4.4 Others: Analysis of Hyperparameters

In Fig. 5, we report the influence of hyperparameters, including parameter τ (in Eq. 3) and batch size. These experiments are carried out on the OpenSSL [50, 200] dataset. In Fig. 5 (a), we report the CGSim performance among a given batch size range. When adding the batch size from 2 to 16, the performance steadily rises. Then we further increase the batch size, and the performance fluctuates. In Fig. 5 (b), we report the effect on parameter τ. There is no doubt that the performance of CGSim improves as the τ value grows until it reaches a figure of 0.7.

after that, it gradually leveled off. As we can see from the figure above, different τ values can have an impact on the model in a significant way. For this reason, we set the $\tau = 0.7$ and batch size as 16 for our model to avoid over-tuning these parameters (such as τ) for various datasets and tasks as well as taking resource consumption into consideration.

5 CONCLUSION

Graph representation learning and cross-graph interaction are pivotal for graph similarity learning. The current researches focus on the cross-graph attention mechanism with high complexity. To improve this situation, in this study, we improve graph similarity learning from two aspects: (1) graph representation learning and (2) cross-graph interaction. For (1), we train the graph representation learning module and the dual contrastive learning module jointly. For (2), our method significantly reduces the time complexity of the proposed node-graph matching mechanism and the graph-graph matching mechanism. These two parts are beneficial to each other. Experimental results show that CGSim effectively enhances the effectiveness and efficiency of graph similarity learning.

ACKNOWLEDGEMENT

The work of S. Pan was supported in part by an ARC Future Fellowship (FT210100097). This work was also supported by the Natural Science Foundation of China under Grants 62272340, and 62276187.

REFERENCES

[1] Rami Al-Rfou, Bryan Perozzi, and Dustin Zelle. 2019. Ddgk: Learning graph representations for deep divergence graph kernels. In *The World Wide Web Conference*. 37–48.

[2] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. 2019. SimGNN: A Neural Network Approach to Fast Graph Similarity Computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, J. Shane Culpepper, Alistair Moffat, Paul N. Bennett, and Kristina Lerman (Eds.). ACM, 384–392. https://doi.org/10.1145/3289600.3290967

[3] Yunsheng Bai, Hao Ding, Ken Gu, Yizhou Sun, and Wei Wang. 2020. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3219–3226.

[4] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence* 7, 04 (1993), 669–688.

[5] Carlota Cardoso, Rita T Sousa, Sebastian Köhler, and Catia Pesquita. 2020. A collection of benchmark data sets for knowledge graph-based similarity in the biomedical domain. *Database* 2020 (2020).

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[7] Yujun Chen, Ke Sun, Juhua Pu, Zhang Xiong, and Xiangliang Zhang. 2020. Grapasa: parametric graph embedding via siamese architecture. *Information Sciences* 512 (2020), 1442–1457.

[8] Keren Fu, Deng-Ping Fan, Ge-Peng Ji, Qijun Zhao, Jianbing Shen, and Ce Zhu. 2021. Siamese network for RGB-D salient object detection and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[9] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. *arXiv preprint arXiv:2104.08821* (2021).

[10] Marco Gori, Marco Maggini, and Lorenzo Sarti. 2005. Exact and approximate graph matching using random walks. *IEEE transactions on pattern analysis and machine intelligence* 27, 7 (2005), 1100–1111.

[11] Marco Gori, Marco Maggini, and Lorenzo Sarti. 2005. The RW2 algorithm for exact graph matching. In *International Conference on Pattern Recognition and Image Analysis*. Springer, 81–88.

[12] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733* (2020).

[13] Juris Hartmanis. 1982. Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Review* 24, 1 (1982), 90.

[14] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*. PMLR, 4116–4126.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9729–9738.

[16] Adel Hlaoui and Shengrui Wang. 2002. A new algorithm for inexact graph matching. In *Object recognition supported by user interaction for service robots*, Vol. 4. IEEE, 180–183.

[17] Bo Jiang, Jin Tang, Xiaochun Cao, and Bin Luo. 2017. Lagrangian relaxation graph matching. *Pattern Recognition* 61 (2017), 255–265.

[18] Di Jin, Cuiying Huo, Chundong Liang, and Liang Yang. 2021. Heterogeneous graph neural network via attribute completion. In *Proceedings of the Web Conference 2021*. 391–400.

[19] Di Jin, Luzhi Wang, Yizhen Zheng, Xiang Li, Fei Jiang, Wei Lin, and Shirui Pan. 2022. CGMN: A Contrastive Graph Matching Network for Self-Supervised Graph Similarity Learning. *arXiv preprint arXiv:2205.15083* (2022).

[20] Di Jin, Zhizhi Yu, Pengfei Jiao, Shirui Pan, Philip S. Yu, and Weixiong Zhang. 2021. A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning. *CoRR* abs/2101.01669 (2021).

[21] Ming Jin, Yizhen Zheng, Yuan-Fang Li, Chen Gong, Chuan Zhou, and Shirui Pan. 2021. Multi-Scale Contrastive Siamese Networks for Self-Supervised Graph Representation Learning. In *IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, Zhi-Hua Zhou (Ed.). ijcai.org, 1477–1483. https://doi.org/10.24963/ijcai.2021/204

[22] Hiroki Kanezashi, Toyotaro Suzumura, Dario Garcia-Gasulla, Min-hwan Oh, and Satoshi Matsuoka. 2018. Adaptive pattern matching with reinforcement learning for dynamic graphs. In *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*. IEEE, 92–101.

[23] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362* (2020).

[24] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

[25] Danai Koutra, Ankur Parikh, Aaditya Ramdas, and Jing Xiang. 2011. Algorithms for graph similarity and subgraph matching. In *Proc. Ecol. Inference Conf*, Vol. 17.

[26] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. 2019. Graph Matching Networks for Learning the Similarity of Graph Structured Objects. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 3835–3845. http://proceedings.mlr.press/v97/li19d.html

[27] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1511.05493

[28] Xiang Ling, Lingfei Wu, Saizhuo Wang, Tengfei Ma, Fangli Xu, Alex X. Liu, Chunming Wu, and Shouling Ji. 2021. Multilevel Graph Matching Networks for Deep Graph Similarity Learning. *IEEE Transactions on Neural Networks and Learning Systems* (2021), 1–15. https://doi.org/10.1109/TNNLS.2021.3102234

[29] Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis. 2021. Anomaly detection on attributed networks via contrastive self-supervised learning. *IEEE transactions on neural networks and learning systems* 33, 6 (2021), 2378–2392.

[30] Guixiang Ma, Nesreen K Ahmed, Theodore L Willke, and S Yu Philip. 2021. Deep graph similarity learning: A survey. *Data Mining and Knowledge Discovery* (2021), 1–38.

[31] Guixiang Ma, Nesreen K Ahmed, Theodore L Willke, Dipanjan Sengupta, Michael W Cole, Nicholas B Turk-Browne, and Philip S Yu. 2019. Deep graph similarity learning for brain data analysis. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2743–2751.

[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[33] Panagiotis Papadimitriou, Ali Dasdan, and Hector Garcia-Molina. 2010. Web graph similarity for anomaly detection. *Journal of Internet Services and Applications* 1, 1 (2010), 19–30.

[34] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[35] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. 2020. SuperGlue: Learning Feature Matching With Graph Neural Networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. IEEE, 4937–4946. https://doi.org/10.1109/CVPR42600.2020.00499

[36] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

[37] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=rJXMpikCZ

[38] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax. *ICLR (Poster)* 2, 3 (2019), 4.

[39] Runzhong Wang, Junchi Yan, and Xiaokang Yang. 2019. Learning Combinatorial Embedding Networks for Deep Graph Matching. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 3056–3065. https://doi.org/10.1109/ICCV.2019.00315

[40] Runzhong Wang, Junchi Yan, and Xiaokang Yang. 2020. Combinatorial Learning of Robust Deep Graph Matching: an Embedding based Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), 1–1. https://doi.org/10.1109/TPAMI.2020.3005590

[41] Runzhong Wang, Junchi Yan, and Xiaokang Yang. 2020. Graduated Assignment for Joint Multi-Graph Matching and Clustering with Application to Unsupervised Graph Matching Network Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/e6384711491713d29bc63fc5eeb5ba4f-Abstract.html

[42] Man Wu, Shirui Pan, and Xingquan Zhu. 2022. Attraction and Repulsion: Unsupervised Domain Adaptive Graph Contrastive Learning Network. *IEEE Transactions on Emerging Topics in Computational Intelligence* (2022).

[43] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2022. Beyond low-pass filtering: Graph convolutional networks with automatic filtering. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[44] Zonghan Wu, Da Zheng, Shirui Pan, Quan Gan, Guodong Long, and George Karypis. 2022. TraverseNet: Unifying Space and Time in Message Passing for Traffic Forecasting. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[45] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=ryGs6iA5Km

[46] Kun Xu, Liwei Wang, Mo Yu, Yansong Feng, Yan Song, Zhiguo Wang, and Dong Yu. 2019. Cross-lingual knowledge graph alignment via graph matching neural network. *arXiv preprint arXiv:1905.11605* (2019).

[47] Xiaojun Xu, Chang Liu, Qian Feng, Heng Yin, Le Song, and Dawn Song. 2017. Neural Network-based Graph Embedding for Cross-Platform Binary Code Similarity Detection. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 363–376. https://doi.org/10.1145/3133956.3134018

[48] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1365–1374.

[49] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems* 33 (2020), 5812–5823.

[50] Tianshu Yu, Runzhong Wang, Junchi Yan, and Baoxin Li. 2020. Learning deep graph matching with channel-independent embedding and Hungarian attention. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=rJgBd2NYPH

[51] Zhizhi Yu, Di Jin, Ziyang Liu, Dongxiao He, Xiao Wang, Hanghang Tong, and Jiawei Han. 2021. AS-GCN: Adaptive Semantic Architecture of Graph Convolutional Networks for Text-Rich Networks. In *ICDM*. IEEE, 837–846.

[52] Laura A Zager and George C Verghese. 2008. Graph similarity scoring and matching. *Applied mathematics letters* 21, 1 (2008), 86–94.

[53] Andrei Zanfir and Cristian Sminchisescu. 2018. Deep Learning of Graph Matching. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2684–2693. https://doi.org/10.1109/CVPR.2018.00284

[54] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. 2021. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 12310–12320. http://proceedings.mlr.press/v139/zbontar21a.html

[55] He Zhang, Bang Wu, Xingliang Yuan, Shirui Pan, Hanghang Tong, and Jian Pei. 2022. Trustworthy Graph Neural Networks: Aspects, Methods and Trends. *arXiv preprint arXiv:2205.07424* (2022).

[56] Yizhen Zheng, Shirui Pan, Vincent Cs Lee, Yu Zheng, and Philip S. Yu. 2022. Rethinking and Scaling Up Graph Contrastive Learning: An Extremely Efficient Approach with Group Discrimination. *CoRR* abs/2206.01535 (2022). https://doi.org/10.48550/arXiv.2206.01535 arXiv:2206.01535

[57] Yizhen Zheng, Yu Zheng, Xiaofei Zhou, Chen Gong, Vincent Lee, and Shirui Pan. 2022. Unifying Graph Contrastive Learning with Flexible Contextual Scopes. *arXiv preprint arXiv:2210.08792* (2022).

[58] Feng Zhou and Fernando De la Torre. 2012. Factorized graph matching. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 127–134.

[59] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep Graph Contrastive Representation Learning. In *Proceedings of the 37th International Conference on Machine Learning Workshop on Graph Representation Learning and Beyond*.