Data Science and Engineering
https://doi.org/10.1007/s41019-024-00278-3

RESEARCH PAPERS

IGFM: An Enhanced Graph Similarity Computation Method with Fine-Grained Analysis

Min Pei[1] · Jianke Yu[2] · Chen Chen[3] · Hanchen Wang[2] · Xiaoyang Wang[4] · Ying Zhang[1]

Received: 29 August 2024 / Revised: 14 November 2024 / Accepted: 15 December 2024

Abstract
In the rapidly advancing field of graph-based applications, accurate graph similarity computing (GSC) has become increasingly important. However, due to the complexity of graph structures, this task remains a challenge because of the intricate calculations involved. To solve the limitations of existing works, this paper introduces the Interpretable Graph Fusion Model (`IGFM`), a novel framework designed to enhance the accuracy and efficiency of graph similarity computation. Specifically, our model can fully utilize graph structure information and comprehensively assess graph similarity at both fine-grained and coarse-grained levels, ultimately achieving more accurate predictions. Experimented extensively across four real-world datasets, `IGFM` demonstrates a significant improvement over existing SOTA methods to solve the GSC challenge. In numerous experimental tests, our model shows performance improvements in terms of MSE (Mean Squared Error), ranging from 4.66% to as much as 56.92% compared to the second-best method.

Keywords Neural network · Graph similarity computation · Graph edit distance · Data mining

1 Introduction

With the rapid development of information technology, a significant number of works have discussed graph structures [1–4]. Graph similarity computation, as one of the most traditional problems in this field, has garnered significant attention in recent works [5, 6]. Nowadays, various applications across different fields are based on graph similarity computation, such as recommendation systems [7], social networks [8–10], computer vision [11], and chemical composition analysis [12]. To adapt to these application scenarios, many diverse and remarkable accurate graph similarity measurement metrics have been proposed, such as Graph Edit Distance (GED) [13], Maximum Common Subgraph [14], Graph Isomorphism [15, 16], *etc*. Specifically, GED is one of the most crucial measure metrics. It quantifies the minimal number of operations necessary to transform one graph into another. It includes modifications to both nodes and edges [17]. Figure 1 shows an example of GED, showing one of the minimum edit distance operations from G_s to G_t. In this toy example, GED computation is achieved through three atomic operations: deleting an edge between nodes 1 and 5, adding an edge between nodes 2 and 5, and modifying the label of node 2. As a result, the GED between G_s and G_t is 3. For simplicity,

✉ Jianke Yu
jianke.yu@student.uts.edu.au

Min Pei
22020100071@pop.zjgsu.edu.cn

Chen Chen
chenc@uow.edu.au

Hanchen Wang
hanchen.wang@uts.edu.au

Xiaoyang Wang
xiaoyang.wang1@unsw.edu.au

Ying Zhang
ying.zhang@zjgsu.edu.cn

[1] Zhejiang Gongshang University, Hangzhou 310018, China

[2] University of Technology Sydney, Ultimo, NSW 2007, Australia

[3] University of Wollongong, Wollongong, NSW 2522, Australia

[4] University of New South Wales, Kensington, NSW 2052, Australia

we represent this process as GED$(G_s, G_t) = 3$. However, the computation of this metric, as well as other accurate metrics, poses substantial computational challenges because it has been proven to be NP-hard [18].

As traditional solutions, combinatorial search-based algorithms are employed to solve GED. These approaches, such as Beam [17], Hungarian [19], VJ [20], calculate GED by pruning infeasible search spaces, establishing lower bounds, and leveraging efficient techniques. While these solutions avoid the complexities associated with NP-hard problems, they still face challenges regarding high computational costs. As learning-based methods have grown, an increasing number of approaches have been proposed to obtain approximate solutions efficiently for the GED. These methods are mainly based on graph neural networks (GNNs) and Neural Tensor Networks (NTN) and typically utilize graph-level interactions to capture the similarity between two graphs [21–24]. Certain models [22–24] have proposed superior strategies for GED prediction based on integrating higher-dimensional embedding at the graph or subgraph level. Moreover, some approaches have further augmented graph-level processing with additional node-level data to refine model performance [21, 25, 26]. Among these works, the Eric model [27] introduced a notably advanced module named Alignment Regularization, which facilitates both node-to-graph and graph-to-graph similarity matching. This technique underscores the significance of fine-grained information, marking a departure from prior learning-based algorithms. Despite these advancements, current models focus predominantly on coarse-grained information and rarely consider fine-grained information to optimize their algorithms. However, node-to-node information is also crucial to compute GED. The model can make prediction easier when node alignment is provided. As a result, existing methods still have space for improvement because they have limitations in using node-to-node information.

Motivated by the above works, we propose a novel model named Interpretable Graph Fusion Model (IGFM). It is a learnable end-to-end model for graph similarity computation. Compared with existing works, the model can better utilize the matching relationships between nodes of two graphs, thereby achieving more excellent predictions of similarity between the graphs. Specifically, IGFM employs three main components: local structure analysis module, fine-grained node-level analysis module, and coarse-grained graph-level analysis module. The local structural analysis module employs cohesive subgraph matching methods (k-core [28], k-truss [29], and clique [30], etc.) to capture the local structural information of connected groups. By discovering these tight groups of nodes, IGFM can capture important details about their local structure, making distinguishing the similarity easier between nodes from different graphs. Then, IGFM utilizes a fine-grained node-level analysis module, which employs the Gumbel-Sinkhorn permutation algorithm and plays a crucial role in facilitating precise node alignment. The result of this fine-grained alignment is crucial for predicting the similarity between two graphs. Inspired by the Neural Tensor Network (NTN) [31] and the Euclidean Distance (ED) [32], the third module of IGFM, named graph-level analysis module, is designed to enhance the model's capability in discerning the differences at the graph level. This module effectively facilitates the computation of a comprehensive similarity assessment for pairs of graphs, thereby bolstering the overall analytical strength of IGFM. The contribution of this work can be summarized as follows:

- To the best of our knowledge, our model is the first to use cohesive subgraph matching techniques to improve GED prediction by capturing fine-grained local structural information. This approach extracts more detailed information for analyzing and computing graph similarities.
- The proposed IGFM demonstrates its capability in balancing fine-grained node-level alignment with coarse-grained graph-level alignment. This dual approach enhances the effectiveness of GED computation simultaneously.
- Comprehensive experimental results obtained from four real-world datasets are presented to validate the effectiveness and efficiency of our model. These experiments demonstrate the superior performance of our model compared to existing methods. The code is published at https://github.com/peimin0815/IGFM.

2 Related Work

In this section, we review significant graph similarity computation works within each category. While there have been extensive studies on graph similarity computation, we categorize the notable works into two main types of methods: (1) combinatorial search-based methods that utilize heuristic

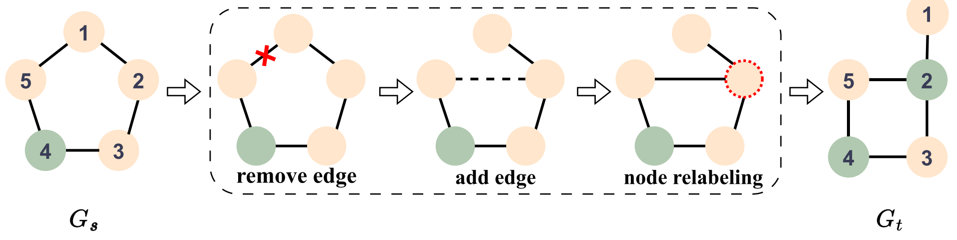

Fig. 1 An example of GED computation. The result of the GED between graphs G_s and G_t is GED$(G_s, G_t) = 3$

searches and lower bound estimations (2) learning-based methods leveraging graph neural networks and neural tensor networks for graph-level interactions. Additionally, we introduce a novel cohesive subgraph mining strategy to enhance graph similarity calculation by identifying dense, interconnected clusters and enhancing node information.

2.1 Graph Similarity Computation

In graph similarity computation, various methods are employed to measure the similarity between graphs. Notable among these methods are Graph Edit Distance (GED) [33], Maximum Common Subgraph (MCS) [34], and Graph Isomorphism [35]. The calculation of these measures is inherently complex and classified as NP-hard [36, 37]. This complexity arises from the combinatorial nature of graph comparison, which requires considering numerous possible mappings between nodes and edges of the graphs. To tackle the computational challenges, various combinatorial search-based methods have been developed. These methods include heuristic searches and lower-bound estimations that aim to approximate the GED efficiently. Some of the notable algorithms in this category are A*-Beam search Algorithm [17]. This algorithm employs a queue of limited size, controlled by a parameter called beam size. It prunes the search space by only keeping the most promising nodes in the queue, thus reducing the computational load. Hungarian Algorithm [19, 38] was originally designed for solving assignment problems. This algorithm has been adapted for graph-matching tasks and provides a way to find the optimal assignment between nodes of two graphs, minimizing the total cost. VJ Algorithm [20] combines ideas from various heuristic methods to improve the efficiency of GED computation. Despite the efficiency improvements these algorithms offer, they still suffer from high time complexity, typically cubic in nature, which limits their scalability to large graphs.

2.2 Learning-Based GED Computation

Recent advancements in deep learning, particularly in Graph Neural Networks (GNNs), have significantly impacted Graph Edit Distance (GED) prediction [39–41, 44]. GNNs, through message passing for node representation learning, effectively capture graph structures and node relationships, enhancing GED computation with remarkable scalability [23, 24, 42, 43]. By training the model with appropriate loss functions and ground-truth GED labels, these models learn embeddings to ensure their distance correlates strongly with GED.

Recent work has addressed special scenarios, such as GED calculation in dynamic graphs [44] and bipartite graphs [45]. In contrast, this work focuses on the traditional GED calculation problem in standard static graphs. SimGNN [21] pioneered the application of GNNs to GED computation. It employs a neural tensor network and a non-differentiable node alignment method to generate graph-level embeddings, simplifying GED computation by outputting a similarity score as the predicted GED. However, SimGNN is limited in that it cannot explicitly predict node-matching relations, reducing its interpretability and fine-grained accuracy. GMN [11] introduces a cross-graph attention layer for interaction between nodes in two graphs, but it only considers graph-level information, resulting in relatively poor prediction accuracy. MGMN [25] introduces a cross-graph matching network that learns interactions between two graphs. It leverages attention mechanisms to combine information at cross-level interactions between each node of one graph, enhancing the accuracy of GED predictions. Despite its remarkable approach, MGMN faces challenges in fine-grained node-level alignments, which are crucial for precise GED calculations.

Additionally, the recent approach EGSC [22] speeds up graph similarity computation by simplifying SimGNN and overlooking node-level interactions, utilizing knowledge distillation to accelerate the inference stage. However, EGSC's omission of cross-graph node-level interactions results in less detailed similarity information, leading to less accurate prediction performance. TaGSim [42] creatively splits GED prediction into predicting the number of each type of graph edit operation, the sum of which is the predicted GED. It proposes a concise network architecture using a graph aggregation layer, improving efficiency and scalability. However, TaGSim's focus on graph-level interactions limits its ability to capture fine-grained node-level information, affecting the accuracy of its predictions. ERIC [27] introduces alignment regularization, which improves similarity computations by effectively aligning nodes and graphs. This model marks a significant advancement in learning-based GED computation but still faces challenges in achieving high accuracy and interpretability, particularly in fine-grained node-level alignments. GedGNN [26] introduces a cross-matrix module that receives node-level embeddings as input, explicitly constructing a node-matching matrix. This approach allows for precise prediction of node-matching relations, addressing a significant limitation of earlier models such as SimGNN and TaGSim. However, despite its accuracy in node matching, GedGNN still requires further improvements in efficiency and scalability to handle large graphs effectively. Despite the advancements made by these models, learning-based GED algorithms still face significant challenges, particularly in achieving high accuracy and interoperability. Fine-grained node-level alignments are crucial for precise GED calculations but are often overlooked by existing models.

To address these challenges, our approach introduces an innovative method employing the Gumbel-Sinkhorn permutation algorithm. This algorithm transforms the non-differentiable node alignment problem into a differentiable permutation problem, making it easier to optimize and improve the overall performance of GED predictions. In summary, while traditional combinatorial search-based methods provide exact solutions to GED at high computational costs, learning-based methods offer scalable and efficient approximations. Our novel approach aims to bridge the gap by enhancing node-level alignments, ultimately leading to more accurate and interpretable GED computations. By employing advanced techniques such as cohesive subgraph mining and the Gumbel-Sinkhorn permutation algorithm, we enhance both the accuracy and efficiency of learning-based GED computation.

2.3 Cohesive Subgraph Mining

Cohesive subgraph mining, a technique for identifying dense, interconnected clusters in graphs, plays a crucial role in enhancing node information by revealing common patterns and characteristics. This field has seen various models, such as k-truss [46, 47], k-core [48], quasi-clique [49], clique [50, 51], (α,β)-core [52] and (k,ω)-core [53]. The k-core of a graph is a subgraph where each node has at least k connections, allowing it to capture densely connected regions effectively. In contrast, methods like k-truss and k-clique also extract cohesive subgraphs, but k-truss requires each edge to participate in a certain number of triangles, and k-clique requires full connectivity, leading to higher computational costs. This work selects the k-core approach as it balances computational efficiency with the ability to capture meaningful structural patterns, which enhances model performance by focusing on nodes within these cohesive regions.

3 Preliminaries and Problem Statement

The graphs mentioned in our study are undirected and unweighted. We consider a graph denoted as $G(V, E, \Lambda)$, where V represents a set of nodes containing n elements, E denotes the set of edges, and Λ represents the set of labels for the graph's nodes. Then, we denote $\boldsymbol{A} \in \{0,1\}^{n\times n}$ as the adjacency matrix of graph G, where each entry $\boldsymbol{A}_{ij}$ is 1 if there is an edge between nodes i and j, and 0 otherwise. This adjacency matrix represents the connections between nodes in G. $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ represents the feature matrix of the graph, where each row corresponds to the d-dimensional feature vector of a node. Among various methods for computing the similarity of graphs, GED is one of the most crucial metrics to measure the similarity of two graphs, which is defined as follows:

Definition 1 (Graph Edit Distance) Given two graphs: a source graph G_s and a target graph G_t. The Graph Edit Distance (GED) between G_s and G_t GED(G_s,G_t) is defined as the minimum number of atomic operations required to transform G_s into G_t. These operations include adding or removing a node, adding or removing an edge, and relabeling a node label.

In this work, the problem is to predict the GED between G_s and G_t. This predicted GED value serves as a measure of the similarity between the two graphs.

Analyzing the structural distribution of two graphs is the key step in calculating graph similarity. In our study, we achieve the goal by searching dense subgraphs for the graphs. Specifically, IGFM chooses k-core criterion to analyze the subgraph density around a certain node, where each node in a k-core connects with at least k other nodes. The definition of k-core is the following:

Definition 2 (k-core) Given a graph $G(V, E)$ and an integer k, the k-core of G is the union of all maximal connected induced subgraphs where each node has a degree of at least k. We denote the k-core as G_k, with node set $V_k \subseteq V$ and edge set $E_k \subseteq E$.

Note that G_k may consist of multiple disconnected components. Figure 2 shows an example to search k-core subgraphs of a graph. We gradually remove nodes with degrees less than 1, obtaining the 1-core subgraph. Then, we delete nodes with a degree less than 2, resulting in the 2-core subgraph. Finally, we follow the same process to get the 3-core subgraph.

After analyzing the density level of nodes by decomposing the graph to k-core subgraph, the coreness of each node can generate its local structure feature.

Definition 3 (coreness) Given a graph $G(V, E)$, the coreness $Core(v)$ of a node $v \in V$ is defined as the largest integer for which the node is included in a $Core(v)$-core.

The local structural feature generated by coreness can give the model a better command of extracting local information from graphs.

Problem Statement. For the graph similarity computation problem, Our objective is to design a learnable end-to-end model for GSC. Specifically, given a pair of graphs G_s and G_t are entered into a well-designed model to predict the GED value between them to represent the similarity between these two graphs.

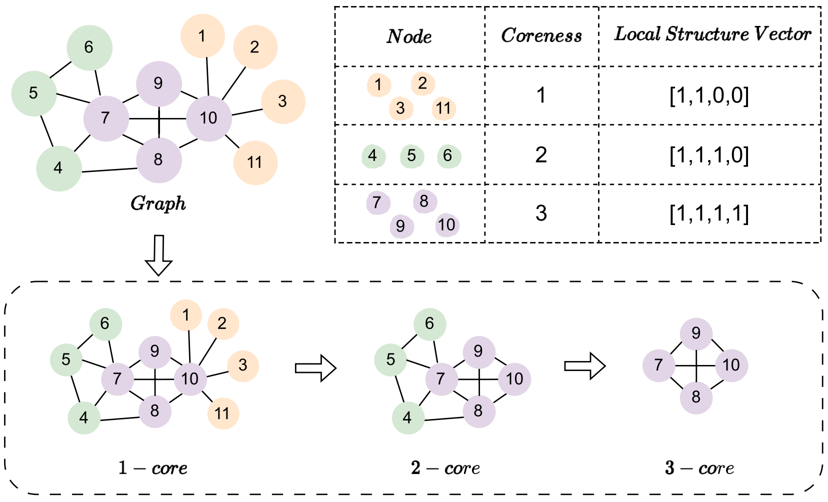

Fig. 2 An example of k-core decomposition and its corresponding local structure vectors

4 Model

Algorithm 1 IGFM Framework

```
   Input: adjacency matrices $\boldsymbol{A}_s$ and $\boldsymbol{A}_t$, and
          node feature matrices $\boldsymbol{X}_s$ and $\boldsymbol{X}_t$ of
          $G_s$ and $G_t$
   Output: a similarity prediction score between
          $G_s$ and $G_t$
1  $\boldsymbol{H}_s^{(0)} \leftarrow X_s$;
2  $\boldsymbol{H}_t^{(0)} \leftarrow X_t$;
3  /* Step 1: Local Structural Analysis Module*/
4  $\bar{\boldsymbol{X}}_s^{(0)} \leftarrow k$-core feature initial $G_s$;
5  $\bar{\boldsymbol{X}}_t^{(0)} \leftarrow k$-core feature initial $G_t$;
6  for $l \in [1, L]$ do
7      $\bar{\boldsymbol{X}}_s^{l} \leftarrow \mathrm{MLP}(\bar{\boldsymbol{X}}_s^{(l-1)})$;
8      $\bar{\boldsymbol{X}}_t^{l} \leftarrow \mathrm{MLP}(\bar{\boldsymbol{X}}_t^{(l-1)})$;
9      $\boldsymbol{H}_s^{l} = \mathrm{GNNEncoder}(\boldsymbol{A}_s, \boldsymbol{H}_s^{(l-1)}, \bar{\boldsymbol{X}}_s^{(i)})$;
10     $\boldsymbol{H}_t^{l} = \mathrm{GNNEncoder}(\boldsymbol{A}_t, \boldsymbol{H}_t^{(l-1)}, \bar{\boldsymbol{X}}_t^{(i)})$;
11 end
12 /* Step 2: Fine-grained Node-level Analysis */
13 $\boldsymbol{P} = \textbf{Gumbel-Sinkhorn}(\boldsymbol{H}_s, \boldsymbol{H}_t)$;
14 NodeSim = $S_n(\boldsymbol{H}_s - \boldsymbol{P}\boldsymbol{H}_t)$;
15 /* Step 3: Coarse-grained Graph-level
     Analysis */
16 $\boldsymbol{Q}_s = \mathrm{Graph\text{-}Pooling}(\boldsymbol{H}_s)$;
17 $\boldsymbol{Q}_t = \mathrm{Graph\text{-}Pooling}(\boldsymbol{H}_t)$;
18 GraphSim = $\alpha\mathrm{NTN}(\boldsymbol{Q}_s, \boldsymbol{Q}_t) + \beta\mathrm{ED}(\boldsymbol{Q}_s, \boldsymbol{Q}_t)$;
19 Score = $\lambda$NodeSim + $(1 - \lambda)$GraphSim;
20 return Score
```

In this section, we introduce the details of IGFM. The model comprises three main parts. The local structural analysis module is presented in Sect. 4.1, followed by the fine-grained node-level analysis module in Sect. 4.2, and lastly, the coarse-grained graph-level analysis module is discussed in Sect. 4.3. Figure 3 shows the model overview, and the framework is illustrated in Algorithm 1. The three main steps of the framework correspond to the three modules. The local structural analysis module is the first step in Lines 4 to 10. For each node, we use k-core to generate local structure information for the node as the enhanced feature. Then, a GNN encoder aggregates node-level embedding. Lines 13 to 14 show the second step. The Gumbel-Sinkhorn Permutation (GSP) algorithm assists the model in capturing the fine-grained node alignment of input graph pairs. Finally, Lines 16 to 18 describe the third step. A multi-scale discriminate mechanism designed to train coarse-grained graph-level similarity. After these steps, the target score used to guide model training is calculated.

4.1 Local Structural Analysis Module

This section primarily focuses on the local structural analysis module, detailing the generation strategy of local structural information for both source and target graphs. Note that the operations performed on both the source graph and the target graph are identical.

First, IGFM calculates the coreness of each node in graphs. It progressively identifies each k-core. For each k, nodes with degrees less than k are iteratively removed until all remaining nodes have at least degree k. Nodes removed during the iteration for a particular k are assigned a coreness of $k - 1$. IGFM continues by increasing k and identifying the next k-core, until no further k-cores can be formed. Figure 2 is an example of calculating coreness for each node. We start by identifying the 2-core subgraph, resulting in the deletion of nodes $\{1, 2, 3, 11\}$, which have a coreness of 1. These nodes have a local structure vector of [1, 1, 0, 0], indicating their presence in the 0-core and 1-core substructures. Next, in the 3-core subgraph, we identify deleted nodes {4, 5, 6} with local structure vectors of [1, 1, 1, 0]. Finally, the remaining nodes {7, 8, 9, 10} have a coreness of 3 and local structure vectors of [1, 1, 1, 1]. Thanks to the analysis strategy, IGFM can effectively extract a d_c-dimensional local structural information $\bar{\boldsymbol{X}}_v$ for each node $v \in V$:

$$\bar{\boldsymbol{X}}_v = [x_i]_{i=1}^{d_c}, \text{ where } x_i = \begin{cases} 1 & \text{if } i \leq \text{Core}(v), \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Finally, we get all local structural feature matrix $\bar{\boldsymbol{X}}_s$ and $\bar{\boldsymbol{X}}_t$ of the source graph G_s and target graph G_t. We employ a Multi-layer Perceptron (MLP) to enhance structural feature matrices for the following modules. This enhancement is crucial for providing a reference for node differentiation, aiding the model in future downstream tasks. The l-th layer of the MLP is operated by the equation $\bar{\boldsymbol{X}}_v^l = \sigma(\bar{\boldsymbol{X}}_v^{(l-1)}\boldsymbol{W}^l + \boldsymbol{b}^l)$, where $\bar{\boldsymbol{X}}_v^{(0)} = \bar{\boldsymbol{X}}_v$, $\boldsymbol{W}^l$ and $\boldsymbol{b}^l$ represent the learnable weight matrix and bias vector of the l-th layer, respectively.

Then, $\bar{\boldsymbol{X}}_v^l$ participates in the encoding process to strengthen the effectiveness of the model. More specifically, we chose the GIN [54] as the encoder. It is known for its ability to represent graph structures. The process of l-th GIN layer is as follows:

$$\boldsymbol{Z}_v^{(l+1)} = \phi^l\left(\left(1+\epsilon^l\right) \cdot \bar{\boldsymbol{X}}_v^l \oplus \boldsymbol{Z}_v^l + \sum_{u \in N_n} \boldsymbol{Z}_u^l\right), \tag{2}$$

where ϵ is a learnable parameter or fixed scalar parameter to control the importance of the node itself, $\bar{\boldsymbol{X}}_v^l$ indicates the local structure feature of node v at l-th MLP layer, notation $\oplus$ denotes concatenating process, $N(v)$ is the set of neighbor nodes of node v, $\boldsymbol{Z}_v^l \in \mathbb{R}^{1\times d^l}$ denotes the feature at the l-th layer and $\boldsymbol{Z}_v^{(0)} = \boldsymbol{X}_v$. With the assistance of the L layer GNN encoder module, node embeddings for G_s and G_t can be obtained as $\boldsymbol{H}_s$ and $\boldsymbol{H}_t$, respectively.

4.2 Fine-Grained Node-Level Analysis

Aligning corresponding nodes in graphs based on structural similarity is crucial for obtaining accurate GED predictions. Therefore, the objective of this module is to generate a reordered adjacency matrix $\boldsymbol{A}'_t$ of G_t, which is optimally aligned with the adjacency matrix $\boldsymbol{A}_s$ of graph G_s. We pad the smaller graphs (A and X) with zeros to ensure the same node number between graph pairs. Once the optimal $\boldsymbol{A}'_t$ is generated, the calculation of the GED between the two graphs can be processed as follows:

$$\text{GED}(G_s, G_t) = |n_s - n_t| + \frac{||\boldsymbol{A}_s - \boldsymbol{A}'_t||_1}{2} + \text{LD}(\Lambda_s, \Lambda_t), \tag{3}$$

where $|\cdot|$ represents absolute value, n_s and n_t are the numbers of two graph nodes, $||\cdot||_1$ represents L1 norm, The function $\text{LD}(\Lambda_s, \Lambda_t)$ quantifies the dissimilarity between the label sets Λ_s of V_s and Λ_t of V_t, measuring how different the labels of corresponding nodes are between the two graphs. However, finding the optimal $\boldsymbol{A}'_t$ is challenging, as node alignment represents a quadratic term assignment problem. It is notoriously difficult to solve within polynomial time constraints [55]. Therefore, we approach this difficulty by searching for a permutation matrix $\boldsymbol{P}$. This matrix is used to reorder the columns and rows of $\boldsymbol{A}_t$ to optimally align it with

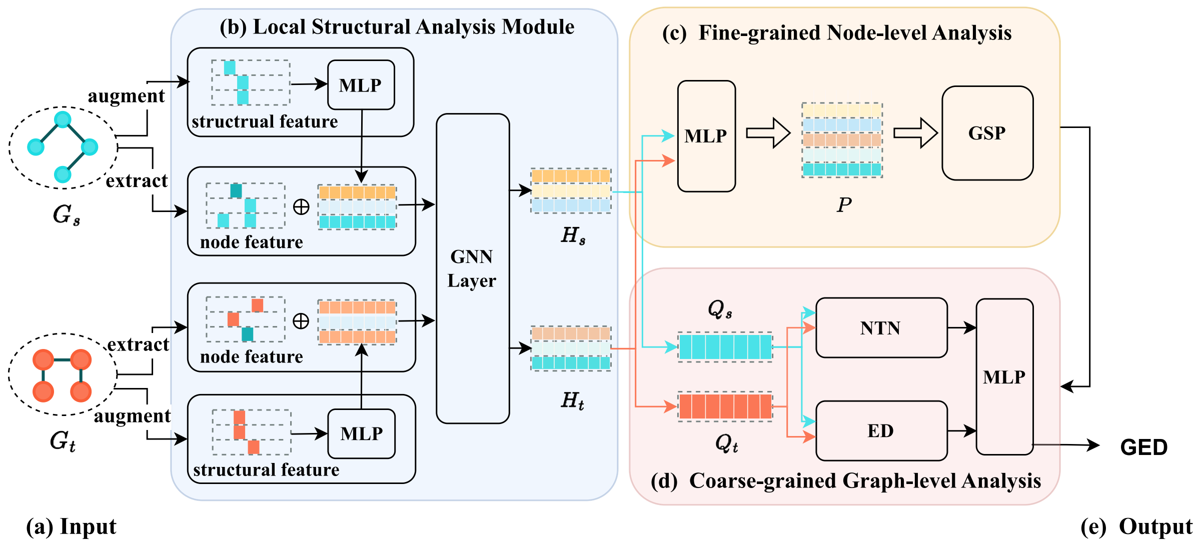

Fig. 3 The overall framework of IGFM

A_s. In other words, the goal of this approach is to search the optimal P and obtain $A'_t = PA_tP^T$. As a result, the optimal P can be denoted as follow:

$$P = \arg\min_{P}(|A_s - PA_tP^T|). \tag{4}$$

It's important to note that P is uniquely structured with exactly 1 in each row and column, and all other elements are 0. Figure 4 shows an example of a node alignment process, where an optimal permutation matrix P is used to transform the adjacency matrix A_t into a new matrix A'_t that closely aligns with A_t. In this example, an additional node is added to graph G_s when searching for the permutation matrix P, to ensure that the matrices A_s and PA_tP^T are of compatible dimensions, allowing for the valid calculation of $|A_s - PA_tP^T|$.

Obtaining an accurate P remains challenging. A naive approach is to employ linear assignment [56]. However, this discrete solution does not support the optimization of learnable models. Inspired by [55], we employ the Gumbel-Sinkhorn permutation (GSP) as a fine-grained learning module to transform it into a continuous method, enabling IGFM to achieve end-to-end implementation. At the same time, each permutation matrix is guaranteed to have a certain randomness by adding noise ζ to H and using δ to control the sensitivity of H_t. Finally, a nearly optimal ordering matrix P is obtained through the normalization of the rows and columns of η times.

Specifically, instead of P, we obtain approximates $\tilde{P}$ by utilizing both the Gumbel-Softmax [57] and the Sinkhorn Operator [55], thereby rendering the node alignment problem learnable. First, we initialize a raw similarity matrix $R \in \mathbb{R}^{n\times n}$, which is constructed to represent the node similarities between two graphs. To compute the matrix R, we use the node feature matrices H_s and H_t of the respective graphs. These matrices are processed through an MLP, and the resulting transformed features are then multiplied as follows:

$$R = \mathrm{MLP}(H_s) \times \mathrm{MLP}(H_t)^T, \tag{5}$$

where the element R_{ij} indicates the similarity between node i in G_s and node j in G_t. Then, GSP calculates the approximate solution $\tilde{P}$ through Gumbel-Softmax and Sinkhorn Operator:

$$\begin{aligned} \tilde{P}^{(0)} &= \exp((\log(R) + \zeta)/\delta), \\ \tilde{P}^{(l+1)} &= \mathcal{N}_{\mathrm{col}}(\mathcal{N}_{\mathrm{row}}(\tilde{P}^{l})), \end{aligned} \tag{6}$$

where ζ is the noise of the Gumbel distribution, parameter δ is utilized to control the sensitivity of GSP during each iteration, $\mathcal{N}_{col}$ and $\mathcal{N}_{row}$ represent normalization operations on the columns and rows of the matrix, respectively. The temperature parameter δ in the GSP algorithm controls the randomness of sampling, with lower values resulting in more deterministic outputs and higher values encouraging exploration of the solution space for effective node alignment in GED calculations. Setting δ involves a trade-off: if δ is too large, the accuracy of the alignment decreases due to excessive randomness; if it's too small, accuracy also suffers because the model lacks sufficient exploration. Therefore, we choose an optimal intermediate value to balance these effects, aiming to achieve the best performance.

After finishing a sufficiently large number of iterations, we can obtain the final $\tilde{P}$ as the approximate solution of P. Finally, we can evaluate the obtained $\tilde{P}$ as follows:

$$S_n(G_s, G_t) = ||H_s - \tilde{P}H_t||_1, \tag{7}$$

where $S_n(G_s, G_t)$ denotes the GSP module prediction score. The smaller this score, the better P we obtained because a smaller score means P provides a better-aligned result. This module can guide IGFM to make full use of fine-grained information.

4.3 Coarse-Grained Graph-Level Analysis

Based on the well-designed fine-grained learning module, IGFM integrates the ability to capture node-level similarities. This capability is essential to improving the model's effectiveness. Building on this foundation, we further

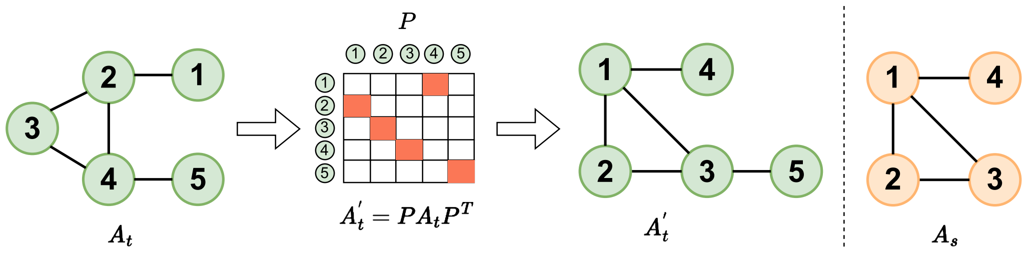

Fig. 4 An example of node alignment

improve the model performance from a graph-level perspective [22, 27]. Specifically, the global coarse-grained graph-level analysis (CGA) module integrates graph-level similarity computation and inter-graph difference computation to achieve the graph-level GED prediction. The core of this approach is to extract a graph-level embedding representation $\boldsymbol{Q}$ for each graph. This representation is obtained through the MLP process as follows:

$$\begin{aligned}\boldsymbol{Q} &= \left[\boldsymbol{Q}^{(1)}\,\boldsymbol{Q}^{(2)}\,\cdots\,\boldsymbol{Q}^{l}\right]\\ \boldsymbol{Q}^{l} &= \mathrm{MLP}_1^{l}\left(\mathrm{Pool}\left(\mathrm{MLP}_2^{l}(\boldsymbol{Z}^{l})\right)\right),\end{aligned} \tag{8}$$

where Pool$(\cdot)$ refers to the column-wise sum pooling function, MLP1^l and MLP2^l are independent MLPs with separate parameters, each corresponding to the l-th layer of the GIN. The pooling operation guarantees the permutation invariance of node representations. Finally, we obtain the graph embedding $\boldsymbol{Q}$ by concatenating the aggregated graph-level information of each hop.

After gaining the graph-level embedding $\boldsymbol{Q}_s$ and $\boldsymbol{Q}_t$ of G_s and G_t, we use the Neural Tensor Network (NTN) [31] module to compute the coarse-grained similarity between graph pairs. NTN model can capture higher-order relationships and patterns in graphs, making it excellent in graph similarity calculation. This module is defined as:

$$S_{ntn}(G_s, G_t) = \sigma(\boldsymbol{Q}_s^T \boldsymbol{W}_n^{[1:H]} \boldsymbol{Q}_t + \boldsymbol{M}_n(\boldsymbol{Q}_s \oplus \boldsymbol{Q}_t) + \boldsymbol{b}_n), \tag{9}$$

where $\boldsymbol{W}_n^{[1:H]}$ is a learnable parameter matrix of the NTN, equipped with a multi-head strategy involving H heads, and $\boldsymbol{M}_n$, $\boldsymbol{b}_n$ are the other learnable parameter matrix and bias of it, and σ is a non-linear activation function.

Furthermore, Euclidean Distance (ED) [32] is utilized to optimize the performance of coarse-grained graph-level analysis. This approach significantly boosts the model's ability to discern differences between similar graphs. The formula for this method can be represented as follows:

$$S_{ed}(G_s, G_t) = \sigma(\exp(-||\boldsymbol{Q}_s - \boldsymbol{Q}_t||_2)), \tag{10}$$

where $||\cdot||_2$ denotes the L2 norm process, exp represents the calculation involving the natural exponential base. This approach guides the model in optimizing parameters by leveraging the difference between two graphs.

After computing both the similarity and difference representations, we can evaluate the similarity between two graphs in a coarse-grained view as follows:

$$S_g(G_s, G_t) = \alpha S_{ntn}(G_s, G_t) + \beta S_{ed}(G_s, G_t), \tag{11}$$

where $S_g(\cdot)$ is the CGA prediction score, α and β are learnable weights that automatically adjust the level of importance the model assigns to the NTN and ED components.

4.4 Model Training

As discussed in Sects. 4.2 and 4.3, the GED prediction problem can be effectively approached by achieving both fine-grained node-level and coarse-grained graph-level analysis.

In the fine-grained node-level analysis, the GED scores for graph pairs G_s and G_t are computed by the GSP module, and their similarity is assessed against ground-truth scores using mean square error (MSE) loss:

$$\mathcal{L}_n = \frac{1}{D} \sum_{(s \times t) \in D} (S_n(G_s, G_t) - S(G_s, G_t))^2, \tag{12}$$

where D represents all training pairs, and $S(\cdot)$ is the ground-truth similarity score normalized from the real GED.

Similarly, for coarse-grained graph-level analysis, the CGA module predicts GED scores, also evaluated using MSE loss:

$$\mathcal{L}_g = \frac{1}{D} \sum_{(s \times t) \in D} (S_g(G_s, G_t) - S(G_s, G_t))^2. \tag{13}$$

The aforementioned two loss functions can be jointly optimized as follows:

$$\mathcal{L} = \lambda \mathcal{L}_n + (1 - \lambda)\mathcal{L}_g, \tag{14}$$

where λ is a tunable hyper-parameter balancing fine and coarse-grained analysis. The learnable parameters of IGFM are optimized using this unified loss function, which maximizes the extraction of graph information for more accurate GED predictions.

5 Experiment

This section thoroughly evaluates our proposed model IGFM on four real-world datasets. We first detail the experimental setup, including dataset descriptions and evaluation metrics. To assess IGFM 's effectiveness and efficiency, it is compared against SOTA baselines. Additionally, we conduct efficiency evaluations, ablation studies, parameter sensitivity analyses, and permutation module analyses, complemented by an in-depth model analysis. These analyses provide crucial insights into each component's role in the model's performance and enhance the model's explainability. To maintain the efficiency of our model, the GSP module will only be utilized in the training phase.

5.1 Experiment Setup

Datasets. We test our model and nine state-of-the-art baselines on four real-world datasets to evaluate both their effectiveness and efficiency. Here is a detailed description of these datasets:

- *LINUX* [21, 58] comprises program dependence graphs derived from the Linux kernel. Each graph corresponds to a function, each node represents a statement, and each edge denotes a dependency. This dataset does not contain the node label set.
- *AIDS700* [21, 59] is a graph of the molecular structure, where the nodes represent the atoms in a molecule, and the edges represent the chemical bonds between the atoms. Each node has a chemical element label indicating its associated properties.
- *IMDB (IMDBMulti)* [60] is derived from the movie information database, which contains multiple relationships between entities such as actors and directors. Each node in the graph represents entities (such as directors or actors), and edges represent relationships between different entities (such as actors in movies). This dataset does not contain the node label set.
- *NCI109* [61] contains 4127 chemical compounds. Their ability to suppress or inhibit human tumor cell growth is tested. Node labels represent the type of atoms, and edges represent chemical bonds.

Table 1 provides an overview of these datasets, detailing the number of graphs, the average number of nodes $|V|$ and edges $|E|$, along with the count of node labels and the number of graph pairs. Consistent with existing studies, the GED calculation process in our work does not take edge labels into account. We split all datasets into training, validation, and testing sets in a 6:2:2 ratio. For the LINUX and AIDS700 datasets, we used the exact A^* algorithm [62] for ground-truth calculations. As previous work [18], no reliable algorithm can efficiently compute GED for graphs with more than 16 nodes, like IMDB and NCI109. In these cases, we considered the minimum values from three approximation algorithms as ground-truth [21].

Baseline methods. In this paper, we mainly compare the state-of-the-art baseline methods into two categories: heuristic approximate GED algorithm and learning-based methods. Heuristic approximate GED methods design an accurate estimation to predict the real GED, including:

- *A*-Beam search (Beam)* [17] Beam is a variant of the A* algorithm that achieves sub-exponential time complexity through Beam search.
- *Hungarian* [19, 38] a cubic time complexity algorithm, is derived from the graph similarity computation problem to solve the bipartite graph matching problem.
- *VJ* [20] is a variant of the Hungarian method.

Table 1 The statistics of the datasets

Dataset	#Graphs	#Avg.$\|V\|$	#Avg.$\|E\|$	#Labels	#Graph pairs
LINUX	1000	7.6	6.9	1	1 M
AIDS700	700	8.9	8.8	29	490 K
IMDB	1500	13.0	65.9	1	2.25 M
NCI109	4127	29.6	32.1	38	17 M

Learning-based approaches aim to predict GED by training well-designed models. Generally, the GNN models are used to fully learn the structure information of graph pairs in the training stage and predict the similarity value of graph pairs in the test stage. In this work, the following learning-based approaches are compared:

- *SimGNN* [21] uses graph embeddings based on the attention mechanism to compute relevance scores.
- *MGMN* [25] employs a node-graph matching network to capture cross-level features between individual nodes of one graph and the entirety of another graph.
- *EGSC* [22] is a graph similarity calculation method based on knowledge distillation that uses a lighter student model to learn from the teacher model.
- *TaGSim* [42] extracts node/edge-level embeddings by graph aggregate layers (GALs) based on the unique transformative impacts of this type of editing.
- *ERIC* [27] uses a novel alignment regularization (AReg) technique, which can learn the optimal alignment information between graphs through node-graph similarity during the training phase.
- *GedGNN* [26] uses a cross-matrix module that captures node-node interaction information between graph pairs.

Note that there are two versions of GedGNN, and we use the more accurate GedGNN-value to compare it with our proposed model. Furthermore, we keep the parameter settings of all learning-based models as they are set in the published paper or optimize the settings to get the best results when the parameters are not provided.

Setting and environment of IGFM. We set the number of GIN [54] layer as 3, the layer of Sinkhorn Operator $L = 20$, a balancing parameter $\lambda = 0.5$ and its temperature $\delta = 0.001$. The model parameters are optimized using the Adam optimizer [63]. Experiments were conducted using PyTorch and PyTorch Geometric on a system with two Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz, NVIDIA RTX 6000 24GB GPU, and 128 G RAM, running Ubuntu 22.04.3 LTS. Core decomposition method [64] is used to obtain the maximum k-core subgraph in our model.

Evaluation metrics. Our model's effectiveness was evaluated using five commonly used graph similarity metrics. The

Mean Squared Error (MSE) measures the squared difference between predicted scores and ground-truth, with lower values indicating more accurate similarity estimates. Spearman's Rank Correlation Coefficient (ρ) and Kendall's Rank Correlation Coefficient (τ) assess the consistency in ranking similarity of nodes in two graphs. Precision at k ($p@k$) evaluates how many top-k results are similar to the query graph. Higher values of ρ, τ, and $p@k$ indicate better performance.

5.2 Performance Evaluation

The experiment results are shown in Tables 2 and 3. The best results are highlighted in bold, and the second-best results are underlined. For simplicity, we scaled the reported MSE values by multiplying them by 10^3 for better visibility and easier comparison.

Generally analyzing these experimental results On these four datasets, the overall performance of IGFM outperforms all other baselines. More detailed, learning-based methods, including IGFM, generally showed superior performance over search-based algorithms. These results indicate that, compared with traditional methods, learning-based algorithms have a powerful ability to analyze complex graph structures. Specifically, IGFM demonstrated improvements over the second-best baseline by more than 27.01%, 14.88%, 4.66%, and 56.92% in MSE across the datasets, and also excelled in global sorting metrics (ρ, τ, $p@10$, $p@20$). Models that focus only on graph-level considerations, including SimGNN, and TagSim, overlook the benefits of node alignment and GED accuracy. As a result, their effectiveness is typically lower than other learning-based methods utilizing more fine-grained information. Due to the MGMN model's excessively dense node-graph computations, it exceeds GPU memory limits on the NCI109 dataset. GedGNN considers node-level information. However, it primarily focuses on generating editable paths. This oversight is the reason why GedGNN's GED predict effectiveness is limited. ERIC is the best baseline now because of its fine-grained analysis ability. However, its node-to-graph approach does not sufficiently consider fine-grained node-level information. This results in lower prediction accuracy than IGFM.

In summary, IGFM exhibited remarkable performance in graph similarity prediction tasks. It surpasses both combinatorial search-based algorithms and the most advanced existing learning-based algorithms.

5.3 Efficiency Evaluation

We further assess the inference time of IGFM to evaluate its efficiency, comparing it against baseline models as depicted in Fig. 5. Since the A* algorithm cannot handle similarity

Table 2 Overall GED performance evaluation on LINUX and AIDS700

	LINUX					AIDS700				
	mse $\downarrow$	$\rho\uparrow$	$\tau\uparrow$	$p@10\uparrow$	$p@20\uparrow$	mse $\downarrow$	$\rho\uparrow$	$\tau\uparrow$	$p@10\uparrow$	$p@20\uparrow$
Beam	9.268	0.827	0.714	0.973	0.924	12.090	0.609	0.463	0.481	0.493
Hungarian	29.805	0.638	0.517	0.913	0.836	25.296	0.510	0.378	0.360	0.392
VJ	63.863	0.581	0.45	0.287	0.251	29.157	0.517	0.383	0.310	0.345
SimGNN	1.017	0.967	0.854	0.942	0.948	3.699	0.816	0.642	0.463	0.546
MGMN	1.564	0.972	0.877	0.968	0.939	2.509	0.897	0.739	0.434	0.507
EGSC	0.319	0.981	0.889	0.980	0.986	1.558	0.895	0.732	0.637	0.715
TaGSim	1.827	0.964	0.844	0.920	0.931	3.682	0.857	0.724	0.659	0.75
ERIC	0.137	**0.987**	0.906	0.983	0.991	1.533	0.891	0.732	0.667	0.737
GedGNN	0.487	0.967	0.906	0.973	0.989	3.06	0.873	0.742	0.586	0.653
IGFM	**0.100**	**0.987**	**0.908**	**0.991**	**0.999**	**1.305**	**0.916**	**0.764**	**0.712**	**0.757**

Table 3 Overall GED performance evaluation on IMDB and NCI109

	IMDB					NCI109				
	mse $\downarrow$	$\rho\uparrow$	$\tau\uparrow$	$p@10\uparrow$	$p@20\uparrow$	mse $\downarrow$	$\rho\uparrow$	$\tau\uparrow$	$p@10\uparrow$	$p@20\uparrow$
SimGNN	1.476	0.884	0.763	0.817	0.833	9.112	0.506	0.387	0.057	0.078
MGMN	13.82	0.816	0.621	0.249	0.307	–	–	–	–	–
EGSC	0.593	**0.936**	**0.826**	0.858	0.868	9.356	0.545	0.414	0.055	0.078
TaGSim	1.465	0.636	0.740	0.803	0.796	6.654	**0.664**	**0.502**	0.023	0.037
ERIC	0.407	0.878	0.773	0.878	0.872	8.690	0.537	0.401	0.083	0.097
GedGNN	1.899	0.691	0.615	0.735	0.736	7.277	0.403	0.287	0.017	0.029
IGFM	**0.388**	0.918	0.806	**0.886**	**0.894**	**2.865**	0.648	0.487	**0.213**	**0.192**

calculations on large graphs within a reasonable time, we only conducted comparisons on LINUX and AIDS700. The results show that IGFM is the most effective model. Specifically, learning-based methods are much more effective than heuristic approximate methods, and the second most effective model is ERIC. However, ERIC's Alignment Regularization module involves a significant computational load, which our alignment algorithm avoids. Therefore, the efficiency of IGFM is higher than ERIC. Compared with the heuristic approximate methods, the inference speed of IGFM is three orders of magnitude or faster than them. Overall, IGFM exceeds all baselines in inference speed while delivering superior accuracy across all datasets.

5.4 Ablation Study

In this section, we evaluate the impact of two crucial components of our IGFM model: the local structural analysis module and the fine-grained node-level analysis module. Since the GED calculation task is a graph-level task, the graph-level analysis is essential for IGFM to predict the GED results. So, our ablation study focuses on analyzing the optimization effects of the above part on the model. More detailed, we compare three additional variant models of IGFM for comparison: IGFM (w/o Core), which omits the local structural analysis module; IGFM (w/o GSP) excludes the fine-grained node-level analysis module compared with IGFM; and IGFM (w/o Both) removes both components.

Table 4 shows the results of the ablation study. It can be observed that the IGFM generally performed better than several variant models on the three metrics we compared while IGFM (w/o Both) maintained the worst effectiveness. The results prove that integrating these two modules into IGFM significantly boosts prediction performance. When these key modules are included, compared with IGFM (w/o Both), IGFM (w/o Core) and IGFM (w/o GSP) demonstrate substantial improvements in predictive accuracy and overall effectiveness. This indicates that the local structural analysis module, as well as the fine-grained node-level analysis module, are essential for enhancing the final performance of the model. Their principal contribution bolstered the model's ability to mine and analyze valuable fine-grained graph information, which is crucial for achieving precise GED predictions. Moreover, it can be observed that IGFM (w/o GSP) is better than IGFM (w/o Core). It suggests that the local structural feature is vital for the GED prediction task. The strategy promotes the model to consider more

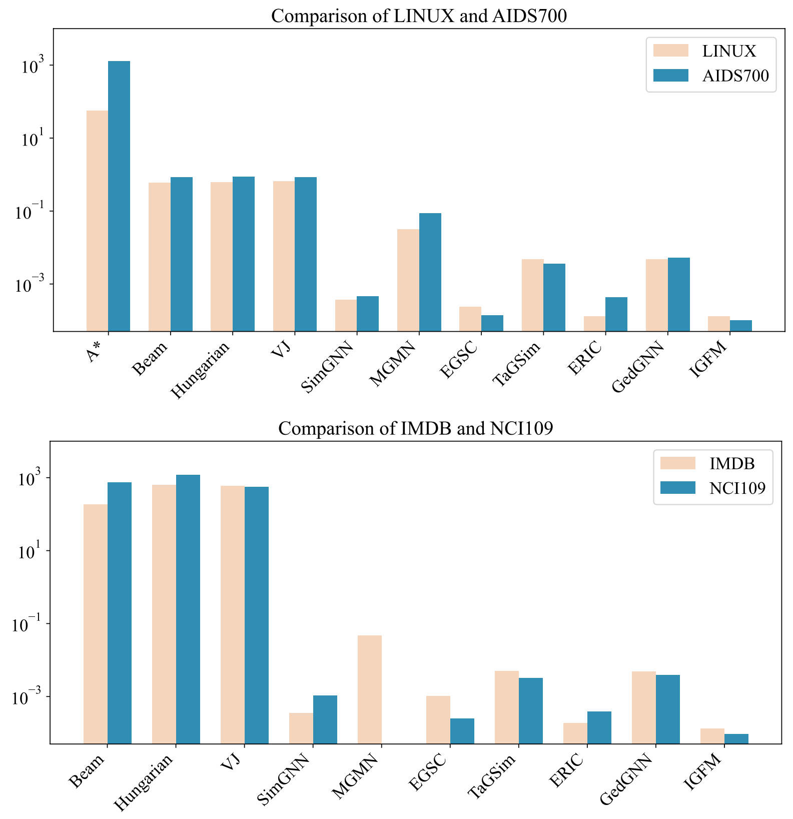

Fig. 5 Evaluation of Inference Time. MGMN ran out of GPU memory on the NCI109 dataset

comprehensive structural information and perform outstanding effectiveness.

5.5 Parameter Sensitivity Analysis

Among the hyper-parameters in our model, the temperature coefficient (δ) is the most important one. That is because it controls the sensitivity of the GSP during each iteration and directly affects the fine-grained node alignment prediction results. We conduct experiments exploring a range of δ sizes ranging from 1 to 1×10^{-4} in decreasing order of magnitude and statistics the model's effectiveness under different δ. The results evaluated by the five metrics are shown in Fig. 6. Obviously, the performance of IGFM is stable when setting different δ. That is because the local structural feature refined the local structure information of each node, making the GSP more stable. Further analyzing the effectiveness of the model under different δ, it can be concluded that when $\delta = 0.001$, the overall prediction performance is the best.

5.6 Permutation Module Analysis

In order to verify the benefits of utilizing the GSP method, we compare the predicted reorder results with the real aligned relationship of a certain graph. More detailed, the trained model is used to predict the reorder results of four randomly selected graph pairs from the LINUX dataset that do not participate in training. Then, we conducted a detailed comparison between the node permutation matrices generated by the GSP and the real aligned relationships.

The results of this analysis are illustrated in Fig. 7. These heatmaps provide a visual representation of the alignment results, with identical numbers used in the node-to-node correspondence matrix. The top row exhibits soft edge alignments generated by GSP, while the bottom row showcases corresponding hard alignments obtained using the Hungarian algorithm. Deeper colors indicate higher similarity levels. It can be concluded that the locations with the highest weights in the GSP-generated permutations closely correspond to those identified by the Hungarian algorithm. These experimental results demonstrate that GSP can accurately

Table 4 The result of ablation study

	LINUX			AIDS700			IMDB			NCI109		
	mse ↓	ρ ↑	τ ↑	mse ↓	ρ ↑	τ ↑	mse ↓	ρ ↑	τ ↑	mse ↓	ρ ↑	τ ↑
IGFM	**0.100**	**0.987**	**0.908**	**1.371**	**0.902**	**0.742**	**0.381**	**0.909**	**0.799**	**2.865**	0.648	0.487
IGFM (w/o Core)	0.137	**0.987**	0.907	1.433	0.897	0.735	0.406	0.866	0.762	6.363	0.585	0.437
IGFM (w/o GSP)	0.120	0.986	0.902	1.376	0.901	0.741	0.384	0.868	0.767	6.009	**0.695**	**0.523**
IGFM (w/o Both)	0.155	0.986	0.902	1.464	0.896	0.735	0.526	0.850	0.742	6.460	0.625	0.465

Bold values indicate the best experimental results, while underlined values highlight the second-best results

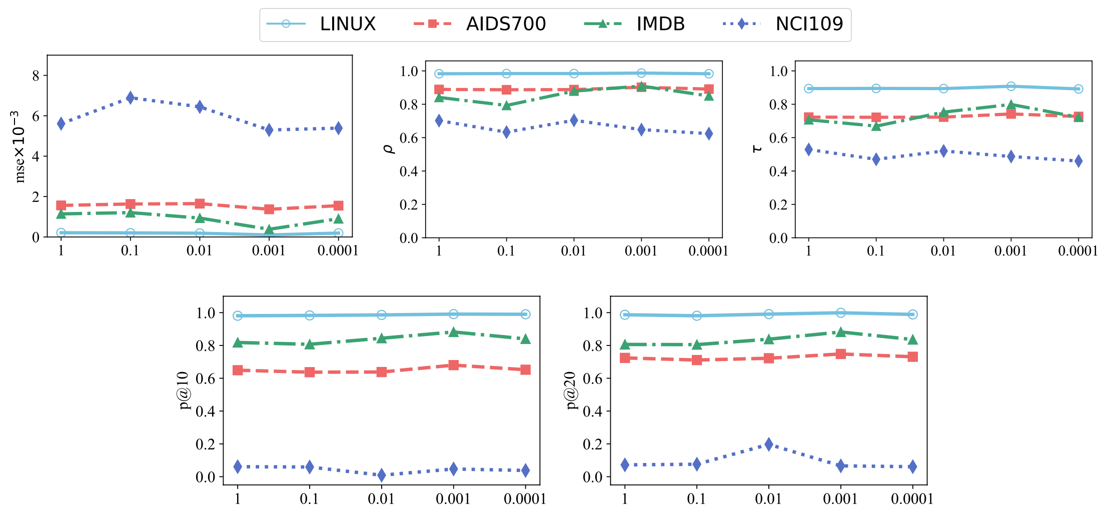

Fig. 6 Impact of different temperature δ on All datasets

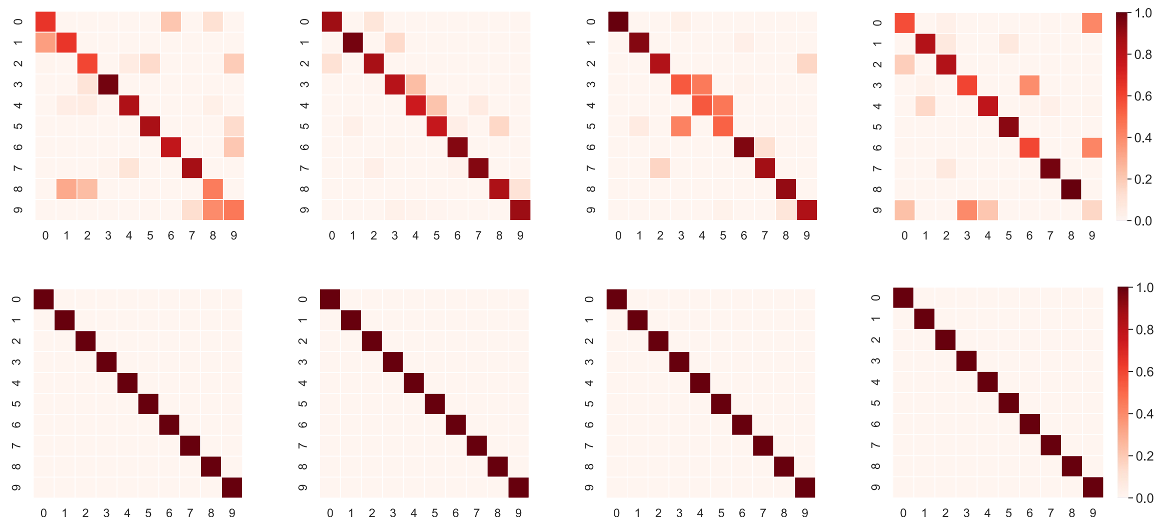

Fig. 7 Analyze the performance of GSP

achieve node alignment operations between graph pairs. This excellent module strongly supports the validity of our model and significantly enhances the GED prediction performance of IGFM.

6 Conclusion

This paper proposes an advanced GED prediction model named IGFM. This model can achieve the essential task of graph similarity computation effectiveness and efficiency. IGFM innovative integrates a local structural analysis module and a coarse-grained graph-level analysis module, while effectively employing the fine-grained node-level analysis module to enhance GED prediction. Our comprehensive evaluations across four real-world datasets demonstrate that IGFM not only makes sense but significantly surpasses the performance of the current SOTA methods, thereby marking a significant advancement in the field of graph similarity computation.

Acknowledgements Not applicable.

Author contributions Min Pei contributed to the conceptualization, writing of the original draft, methodology, software development,and investigation.Jianke Yu was responsible for the conceptualization, writing - review & editing, methodology, and supervision.Chen Chen provided resources, contributed to writing - review & editing, and performed formal analysis.Hanchen Wang handled data curation and participated in writing - review & editing.XiaoyangWang was involved in the conceptualization, writing - review & editing, and project administration.Ying Zhang contributed to funding acquisition and supervision.

Funding Not applicable.

Data availability The GED datasets are available at https://pytorch-geometric.readthedocs.io/, and the code is published at https://github.com/peimin0815/IGFM.

Declarations

Conflict of interests The authors declare that they have no competing interests.

References

1. Zhao X, Xiao C, Lin X, Liu Q, Zhang W (2013) A partition-based approach to structure similarity search. Proc VLDB Endow 7(3):169–180
2. Zhang T, Gao Y, Zheng B, Chen L, Wen S, Guo W (2020) Towards distributed node similarity search on graphs. World Wide Web 23:3025–3053
3. Jepsen TS, Jensen CS, Nielsen TD (2020) Relational fusion networks: graph convolutional networks for road networks. IEEE Trans Intell Transp Syst 23(1):418–429
4. Zhang Y, Cheung WK, Liu Q, Wang G, Yang L, Liu L (2024) Towards explaining graph neural networks via preserving prediction ranking and structural dependency. Inf Process Manag 61(2):103571
5. Bai Y, Ding H, Gu K, Sun Y, Wang W (2020) Learning-based efficient graph similarity computation via multi-scale convolutional

set matching. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 3219–3226
6. Roy I, Velugoti VSBR, Chakrabarti S, De A (2022) Interpretable neural subgraph matching for graph retrieval. In: Proceedings of the AAAI conference on artificial intelligence, vol 36, pp 8115–8123
7. Wang S, Hu L, Wang Y, He X, Sheng QZ, Orgun MA, Cao L, Ricci F, Philip SY (2021) Graph learning based recommender systems: a review. In: 30th international joint conference on artificial intelligence, IJCAI 2021, pp 4644–4652
8. Wang Y, Cong G, Song G, Xie K (2010) Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1039–1048
9. Liu G, Liu Y, Zheng K, Liu A, Li Z, Wang Y, Zhou X (2017) Mcs-gpm: multi-constrained simulation based graph pattern matching in contextual social graphs. IEEE Trans Knowl Data Eng 30(6):1050–1064
10. Gong X, Wang H, Wang X, Chen C, Zhang W, Zhang Y (2024) Influence maximization on hypergraphs via multi-hop influence estimation. Inf Process Manag 61(3):103683
11. Li Y, Gu C, Dullien T, Vinyals O, Kohli P (2019) Graph matching networks for learning the similarity of graph structured objects. In: International conference on machine learning. PMLR, pp 3835–3845
12. Ma G, Ahmed NK, Willke TL, Yu PS (2021) Deep graph similarity learning: a survey. Data Min Knowl Disc 35:688–725
13. Bunke H, Allermann G (1983) Inexact graph matching for structural pattern recognition. Pattern Recogn Lett 1(4):245–253
14. Bunke H, Shearer K (1998) A graph distance metric based on the maximal common subgraph. Pattern Recogn Lett 19(3–4):255–259
15. Dijkman R, Dumas M, García-Bañuelos L (2009) Graph matching algorithms for business process model similarity search. In: Business process management: 7th international conference, BPM 2009, Ulm, Germany, September 8–10, 2009. Proceedings 7. Springer, pp 48–63
16. Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY (2020) A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst 32(1):4–24
17. Neuhaus M, Riesen K, Bunke H (2006) Fast suboptimal algorithms for the computation of graph edit distance. In: Structural, syntactic, and statistical pattern recognition: joint IAPR international workshops, SSPR 2006 and SPR 2006, Hong Kong, China, August 17–19, 2006. Proceedings. Springer, pp 163–172
18. Blumenthal DB, Gamper J (2020) On the exact computation of the graph edit distance. Pattern Recogn Lett 134:46–57
19. Kuhn HW (2005) The Hungarian method for the assignment problem. Naval Res Logist 52(1):7–21
20. Fankhauser S, Riesen K, Bunke H (2011) Speeding up graph edit distance computation through fast bipartite matching. In: Graph-based representations in pattern recognition: 8th IAPR-TC-15 international workshop, GbRPR 2011, Münster, Germany, May 18–20, 2011. Proceedings 8. Springer, pp 102–111
21. Bai Y, Ding H, Bian S, Chen T, Sun Y, Wang W (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, pp 384–392
22. Qin C, Zhao H, Wang L, Wang H, Zhang Y, Fu Y (2021) Slow learning and fast inference: efficient graph similarity computation via knowledge distillation. Adv Neural Inf Process Syst 34:14110–14121
23. Zhang Z, Bu J, Ester M, Li Z, Yao C, Yu Z, Wang C (2021) H2mn: Graph similarity learning with hierarchical hypergraph matching networks. In: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, pp 2274–2284
24. Ranjan R, Grover S, Medya S, Chakaravarthy V, Sabharwal Y, Ranu S (2022) Greed: a neural framework for learning graph distance functions. Adv Neural Inf Process Syst 35:22518–22530
25. Ling X, Wu L, Wang S, Ma T, Xu F, Liu AX, Wu C, Ji S (2021) Multilevel graph matching networks for deep graph similarity learning. IEEE Trans Neural Netw Learn Syst
26. Piao C, Xu T, Sun X, Rong Y, Zhao K, Cheng H (2023) Computing graph edit distance via neural graph matching. Proc VLDB Endow 16(8):1817–1829
27. Zhuo W, Tan G (2022) Efficient graph similarity computation with alignment regularization. Adv Neural Inf Process Syst 35:30181–30193
28. Seidman SB (1983) Network structure and minimum degree. Soc Netw 5(3):269–287
29. Cohen J (2008) Trusses: cohesive subgraphs for social network analysis. Natl Sec Agency Tech Rep 16(3.1):1–29
30. Luce RD, Perry AD (1949) A method of matrix analysis of group structure. Psychometrika 14(2):95–116
31. Socher R, Chen D, Manning CD, Ng A (2013) Reasoning with neural tensor networks for knowledge base completion. Adv Neural Inf Process Syst 26
32. Krislock N, Wolkowicz H (2012) Euclidean distance matrices and applications. Springer, Berlin
33. Sanfeliu A, Fu K-S (1983) A distance measure between attributed relational graphs for pattern recognition. IEEE Trans Syst Man Cybern 3:353–362
34. Bunke H (1997) On a relation between graph edit distance and maximum common subgraph. Pattern Recogn Lett 18(8):689–694
35. Corneil DG, Gotlieb CC (1970) An efficient algorithm for graph isomorphism. J ACM 17(1):51–64
36. Fernández M-L (2001) Valiente G: a graph distance metric combining maximum common subgraph and minimum common supergraph. Pattern Recogn Lett 22(6–7):753–758
37. Chen L, Gao Y, Li X, Jensen CS, Chen G (2015) Efficient metric indexing for similarity search. In: 2015 IEEE 31st international conference on data engineering. IEEE, pp 591–602
38. Riesen K, Bunke H (2009) Approximate graph edit distance computation by means of bipartite graph matching. Image Vis Comput 27(7):950–959
39. Duan Y, Liu J, Chen S, Chen L, Wu J (2024) G-prompt: graphon-based prompt tuning for graph classification. Inf Process Manag 61(3):103639
40. Shang B, Zhao Y, Liu J (2024) Knowledge graph representation learning with relation-guided aggregation and interaction. Inf Process Manag 61(4):103752
41. Liu Z, Meng L, Sheng QZ, Chu D, Yu J, Song X (2024) Poi recommendation for random groups based on cooperative graph neural networks. Inf Process Manag 61(3):103676
42. Bai J, Zhao P (2021) Tagsim: type-aware graph similarity learning and computation. Proc VLDB Endow 15(2)
43. Jin D, Wang L, Zheng Y, Li X, Jiang F, Lin W, Pan S (2022) Cgmn: a contrastive graph matching network for self-supervised graph similarity learning. arXiv preprint arXiv:2205.15083
44. Wang R, Zhang T, Yu T, Yan J, Yang X (2021) Combinatorial learning of graph edit distance via dynamic embedding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 5241–5250
45. Riesen K, Bunke H (2015) Improving bipartite graph edit distance approximation using various search strategies. Pattern Recognit 48(4):1349–1363

46. Zhang F, Zhang W, Zhang Y, Qin L, Lin X (2017) Olak: an efficient algorithm to prevent unraveling in social networks. Proc VLDB Endow 10(6):649–660
47. Zhang F, Zhang Y, Qin L, Zhang W, Lin X (2018) Efficiently reinforcing social networks over user engagement and tie strength. In: 2018 IEEE 34th international conference on data engineering (ICDE). IEEE, pp 557–568
48. Linghu Q, Zhang F, Lin X, Zhang W, Zhang Y (2020) Global reinforcement of social networks: The anchored coreness problem. In: Proceedings of the 2020 ACM SIGMOD international conference on management of data, pp 2211–2226
49. Matsuda H, Ishihara T, Hashimoto A (1999) Classifying molecular sequences using a linkage graph with their pairwise similarities. Theor Comput Sci 210(2):305–325
50. Sun R, Chen C, Wang X, Zhang Y, Wang X (2020) Stable community detection in signed social networks. IEEE Trans Knowl Data Eng 34(10):5051–5055
51. Sun R, Zhu Q, Chen C, Wang X, Zhang Y, Wang X (2020) Discovering cliques in signed networks based on balance theory. In: Database systems for advanced applications: 25th international conference, DASFAA 2020, Jeju, South Korea, September 24–27, 2020, Proceedings, Part II 25. Springer, pp 666–674
52. Yu J, Wang H, Wang X, Li Z, Qin L, Zhang W, Liao J, Zhang Y (2023) Group-based fraud detection network on e-commerce platforms. In: Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining, pp 5463–5475
53. Liu X, Wang X (2021) Cohesive subgraph identification in weighted bipartite graphs. Appl Sci 11(19):9051
54. Xu K, Hu W, Leskovec J, Jegelka S (2018) How powerful are graph neural networks? In: International conference on learning representations
55. Mena G, Belanger D, Linderman S, Snoek J (2018) Learning latent permutations with gumbel-sinkhorn networks. In: International conference on learning representations, vol 2018
56. Bruff D (2005) The assignment problem and the Hungarian method. Notes Math 20(29–47):5
57. Jang E, Gu S, Poole B (2016) Categorical reparameterization with gumbel-softmax. In: International conference on learning representations
58. Wang X, Ding X, Tung AK, Ying S, Jin H (2012) An efficient graph indexing method. In: 2012 IEEE 28th international conference on data engineering. IEEE, pp 210–221
59. Liang Y, Zhao P (2017) Similarity search in graph databases: a multi-layered indexing approach. In: 2017 IEEE 33rd international conference on data engineering (ICDE). IEEE, pp 783–794
60. Yanardag P, Vishwanathan S (2015) Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1365–1374
61. Wale N, Watson IA, Karypis G (2008) Comparison of descriptor spaces for chemical compound retrieval and classification. Knowl Inf Syst 14:347–375
62. Riesen K, Emmenegger S, Bunke H (2013) A novel software toolkit for graph edit distance computation. In: Graph-based representations in pattern recognition: 9th IAPR-TC-15 international workshop, GbRPR 2013, Vienna, Austria, May 15–17, 2013. Proceedings 9. Springer, pp 142–151
63. Kingma D, Ba J (2015) Adam: A method for stochastic optimization. In: International conference on learning representations (ICLR)
64. Malliaros FD, Giatsidis C, Papadopoulos AN, Vazirgiannis M (2020) The core decomposition of networks: theory, algorithms and applications. VLDB J 29(1):61–92