

MATA*: Combining Learnable Node Matching with A* Algorithm for Approximate Graph Edit Distance Computation

Junfeng Liu*

Beihang University
Beijing, China

liujunfeng@buaa.edu.cn

Min Zhou

Huawei Noah's Ark Lab
Shenzhen, China

zhoumin27@huawei.com

Shuai Ma

Beihang University
Beijing, China

mashuai@buaa.edu.cn

Lujia Pan

Huawei Noah's Ark Lab
Shenzhen, China

panlujia@huawei.com

ABSTRACT

Graph Edit Distance (GED) is a general and domain-agnostic metric to measure graph similarity, widely used in graph search or retrieving tasks. However, the exact GED computation is known to be NP-complete. For instance, the widely used A* algorithms explore the entire search space to find the optimal solution which inevitably suffers scalability issues. Learning-based methods apply graph representation techniques to learn the GED by formulating a regression task, which can not recover the edit path and lead to inaccurate GED approximation (*i.e.*, the predicted GED is smaller than the exact). To this end, in this work, we present a data-driven hybrid approach MATA* for approximate GED computation based on Graph Neural Networks (GNNs) and A* algorithms, which models from the perspective of learning to match nodes instead of directly regressing GED. Specifically, aware of the structure-dominant operations (*i.e.*, node and edge insertion/deletion) property in GED computation, a structure-enhanced GNN is firstly designed to jointly learn local and high-order structural information for node embeddings for node matchings. Second, top- k candidate nodes are produced via a differentiable top- k operation to enable the training for node matchings, which is adhering to another property of GED, *i.e.*, multiple optimal node matchings. Third, benefiting from the candidate nodes, MATA* only performs on the promising search directions, reaching the solution efficiently. Finally, extensive experiments show the superiority of MATA* as it significantly outperforms the combinatorial search-based, learning-based and hybrid methods and scales well to large-size graphs.

CCS CONCEPTS

- **Mathematics of computing** → **Approximation algorithms**;
- **Computing methodologies** → *Supervised learning*; Discrete space search.

KEYWORDS

combinatorial optimization; graph edit distance; graph neural networks; node matching; A* algorithm

*This work is mainly done during an internship at Huawei.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0124-5/23/10...\$15.00

<https://doi.org/10.1145/3583780.3614959>

ACM Reference Format:

Junfeng Liu, Min Zhou, Shuai Ma, and Lujia Pan. 2023. MATA*: Combining Learnable Node Matching with A* Algorithm for Approximate Graph Edit Distance Computation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*, October 21–25, 2023, Birmingham, United Kingdom. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3583780.3614959>

1 INTRODUCTION

Graph-structured data are ubiquitous, ranging from chemical compounds [8], social networks [16], computer vision [35] to programming languages [23]. A recurrent and pivotal task when working with these graph-structured applications is assessing how similar or different two given graphs are, among which graph edit distance (GED) is a widely used metric due to its flexible and domain-agnostic merits [3, 9, 10, 23]. In general, GED computation refers to finding the minimum cost of edit operations (*node insertion/deletion, edge insertion/deletion, and node/edge relabeling*) to transform the source graph to a target one [5] (see Fig. 1 for an example).

The exact GED computation guarantees optimality which is however NP-complete [5]. It typically treats all possible edit operations as a pathfinding problem where A* algorithm (a best-first search method) is widely used to expand the search [9–11, 18]. These solutions mainly focus on pruning unpromising search spaces using A* algorithm or filtering dissimilar graph pairs to speed up GED computation. However, they all run in factorial time in the worst case due to the exhaustiveness of their search spaces, such that they cannot reliably compute the GED of graphs with more than 16 nodes in a reasonable time [7].

Some recent works for the approximate GED computation have been proposed with the help of the graph representation techniques, which can be divided into two main categories: Learning-based models [2–4, 23, 26, 27, 41, 42] and hybrid approaches [33, 37]. For learning-based models, they directly formulate the approximate GED computation as a regression task and supervisedly learn the GED as a graph similarity metric in an end-to-end manner. Although such learning-based methods alleviate the computational burden of GED, they could encounter the *inaccurate GED approximation* issue (*i.e.*, the predicted GED is smaller than the exact result) and also fail to recover an actual edit path, which is indispensable in specific tasks *e.g.*, network alignment [21], graph matching [12, 33]. For hybrid methods, deep learning and combinatorial-search techniques are combined to optimize the GED computation. Recently [33] and [37] separately propose two hybrid approaches, both of which apply Graph Neural Networks (GNNs) to guide the search directions of A* algorithms. However, the solved edit distance is typically provided with a large gap, due to their accumulation of the *inaccurate GED approximation* in the cost function estimation

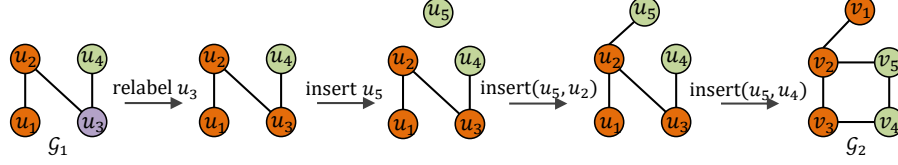


Figure 1: An edit path from source graph \mathcal{G}_1 to target graph \mathcal{G}_2 . Different colors represent the nodes with different labels. Assume the edit costs are uniform, and $\text{ged}(\mathcal{G}_1, \mathcal{G}_2) = 4$. That is at least four edit operations are required to transform \mathcal{G}_1 to \mathcal{G}_2 , where the node mapping corresponding to the edit path is $\{u_1, u_2, u_3, u_4\}$ to $\{v_1, v_2, v_3, v_4\}$. (1) Essentially, there are *two optimal node matchings* for $\text{ged}(\mathcal{G}_1, \mathcal{G}_2) = 4$, and another node mapping is $\{u_1, u_2, u_3, u_4\}$ to $\{v_2, v_3, v_4, v_5\}$. (2) Among the edit operations, there includes one attribute operation (i.e., relabel u_3) and *three structure operations*.

(i.e., the cost of unmatched subgraphs) of A* algorithms. Besides, GNNs with the attention mechanism are employed to estimate the cost function, which take $O(n^2d + d^2n)$ time for extending each search, and encounter scalability issues [33, 37].

It is known that GED computation equals finding the optimal node matching between the source and the target graphs. Once the node matchings are given, GED can be easily calculated by scanning the two graphs once [9], which reveals the intrinsic connection between GED computation and node matching. Besides, existing learning-based and hybrid approaches only formulate GED as a regression task for graph or subgraph pairs, which fails to explicitly consider the node matching in their models. Be aware of the intrinsic connection between GED computation and node matching, in this work, we attempt to learn the node matching corresponding to GED using GNNs. However, it is not trivial as the following two combinatorial properties essentially exist in GED computation. (1) *Multiple optimal node matchings* (i.e., different matchings to produce GED) makes it difficult to learn the node matching by directly modeling in end-to-end learning. (2) *Structure-dominant operations* (i.e., most edit operations are involved in structure) create challenges for incorporating structural information into learning models. Also, see Fig. 1 for an example.

To this end, in this work, we present a data-driven hybrid approach **MATA*** based on Graph Neural Networks and A* algorithms, which leverages the learned candidate **matching** nodes to prune unpromising search directions of A* algorithm (i.e., A*LSa [9]) for approximate GED computation.

Contributions. The main contributions are as follows.

- (1) We present a hybrid approach based on GNNs and A* algorithms rather than via an end-to-end manner, which models GED computation from the perspective of node matching and combines the intrinsic connection between GED computation and node matching.
- (2) A structure-enhanced Graph Neural Network (i.e., SEGNN) is introduced to jointly learn local and high-order structural information for node embeddings *w.r.t.* node matchings from a fine granularity, which captures the combinatorial property of structure-dominant operations in GED computation.
- (3) Further, top- k candidate nodes are produced to be aware of the multiple optimal node matchings combinatorial property via a differentiable top- k operation, which is built upon two complementary learning tasks to jointly generate the candidate nodes, i.e., learning node matching and learning GED.
- (4) We conduct extensive experiments on real-life datasets AIDS,

IMDB, and CANCER to demonstrate the superiority and scalability of MATA* from three types of methods: combinatorial search-based, learning-based and hybrid approaches. Indeed, MATA* improves the accuracy by (39.0%, 21.6%, 11.7%) and reduces the average discrepancy by (6.5%, 9.1%, 24.5%) at least on three real-life datasets (AIDS, IMDB, CANCER), respectively.

2 RELATED WORKS

Computing the graph edit distance between graphs is a classical and fundamental combinatorial optimization problem over graphs where a vast body of literature exists in various domains [2, 4, 9, 10, 15, 18, 23, 25, 27, 28, 30, 33, 37, 41, 42]. We next present a detailed overview of existing literature from three categories: (1) combinatorial search-based, (2) learning-based and (3) hybrid graph edit distance computation.

Combinatorial search-based. Combinatorial search-based algorithms either directly explore the search space corresponding to GED, or relax it to other combinatorial problems with polynomial time complexity. (1) The solution space of exact GED is typically treated as a pathfinding problem where best-search (A* [17, 30]) and depth-first search [1, 6] are utilized to expand the search path [37]. Different exact algorithms mainly focus on how to better estimate the cost of unmatched subgraphs with the theoretical guarantee to prune the search space, such as using label sets [29, 30], and subgraph structures [9, 10, 18]. (2) The approximate algorithms are proposed to find the sub-optimal solutions. [25] explores the most possible directions with the limited beam size of A* algorithms. [28] and [15] only consider the local structure and relax it to bipartite matching problems, which are computed in cubic time.

Learning-based GED computation. With the progress of graph representation techniques of Graph Neural Networks [14, 19, 39], some works directly model it as a regression problem and learn the approximate GED via an end-to-end manner by treating GED as a similarity score between graphs. Different learning-based algorithms mainly focus on designing different GNN models for the graph edit distance computation task. [3] first presents a model using GCN [19] and attention mechanism to approximately learn GED in an end-to-end fashion. Based on [3], [4] further introduces a multi-scale node comparison technique to extract the fine-grained information from the node-to-node similarity matrix. Besides, [23] incorporates both the node and graph level information by the cross-graph module to trade-off the accuracy and computation. [2] splits the graph edit distance into different types of edit operations

and applies graph aggregation layers to learn each type individually. More recently, [26] designs a GED-specific regularizer to impose the matching constraints involved in GED, where the graph pairs are represented by the association graphs. [27] designs a novel siamese graph neural network, which through a carefully crafted inductive bias, learns the graph and subgraph edit distances via a property-preserving manner.

Hybrid GED computation. Recently, there has been a surge of interest in marrying learning-based approaches with combinatorial-search techniques. This interdisciplinary blend has given birth to several hybrid methodologies, particularly those that integrate Graph Neural Networks (GNNs) with the A* search algorithm, as seen in references [33, 37]. Both methods leverage machine learning techniques to enhance the performance of A* algorithms for GED computation, by predicting the cost of unmatched subgraphs to optimize their search directions. [37] proposes graph path networks incorporating pre-training edit path information and cross-graph information for training the model and [33] integrates a dynamic graph embedding network [3] for estimating the cost associated with unmatched subgraphs.

3 PRELIMINARIES

We focus the discussions on the labeled and undirected simple graphs, that is a graph denoted by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \Phi\}$, where \mathcal{V} is the set of nodes, \mathcal{E} is the set of undirected edges with $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and Φ is a label function that assigns labels to each node or edge.

Graph Edit Distance (GED). The graph edit distance between graphs \mathcal{G}_1 and \mathcal{G}_2 is defined as the minimum cost of edit operations (*i.e.*, node insertion/deletion, edge insertion/deletion, and node/edge relabeling) to transform \mathcal{G}_1 to \mathcal{G}_2 , denoted by $\text{ged}(\mathcal{G}_1, \mathcal{G}_2)$ [2, 10, 37]. One specific constraint to note is the aspect of *node deletion* operation, it's restricted only to the nodes that are isolated, ensuring structural integrity and meaningful transformations between the graphs. Due to the NP-completeness of graph edit distance, the approximate edit distance is often used, denoted by $\text{ged}(\mathcal{G}_1, \mathcal{G}_2)$, offers a balance between computational scalability and accuracy. In this work, we focus on the line of uniform edit cost, *i.e.*, all of the edit operations share the same cost [2–4, 9, 10, 26, 27, 37], yet the techniques presented in the following sections can also be extended to handle the non-uniform edit cost.

GED computation from node matchings. We next illustrate how to compute GED from the view of node matchings. Here, the node matching refers to an injective function from the nodes \mathcal{V}_1 to the nodes \mathcal{V}_2 .

Proposition 1: *The ged between the graph pair \mathcal{G}_1 and \mathcal{G}_2 equals the minimum edit cost among all node matchings from the source graph \mathcal{G}_1 to the target graph \mathcal{G}_2 [9].* \square

By proposition 1, the $\text{ged}(\mathcal{G}_1, \mathcal{G}_2)$ can be determined by exhaustively generating all possible matchings from \mathcal{V}_1 to \mathcal{V}_2 , *i.e.*, this essentially translates to identifying the node matching that incurs the least edit cost. Based on the commutativity of $\text{ged}(\cdot, \cdot)$ and the edit cost are uniform, *w.l.o.g.* for a graph pair \mathcal{G}_1 and \mathcal{G}_2 , \mathcal{G}_1 always refers to the graph with fewer nodes in later sections.

Moreover, the optimal node matching between \mathcal{G}_1 and \mathcal{G}_2 can be formulated as the constrained binary quadratic programming

Table 1: Statistics of types of edit operations. We randomly sample 1,000 graphs for each dataset and compute their edit operations, where NI, EI and ED stand for node insertions, edge insertions and edge deletions, respectively.

Datasets	Structure Operations			Attribute Operations
	NI	EI	ED	Relabeling
AIDS	18.3%	34.8%	8.7%	38.0%
IMDB	12.1%	61.8%	5.2%	0.0%
CANCER	4.6%	40.8%	35.5%	18.6%

(CBQP) problem [26]:

$$\begin{aligned}
 \min_X \text{dist} &= \sum_{\substack{u_i \in \mathcal{G}_1 \\ v_k \in \mathcal{G}_2}} c_{i,k} X_{i,k} + \sum_{\substack{u_i, u_j \in \mathcal{G}_1 \\ v_k, v_l \in \mathcal{G}_2}} c_{i,k,j,l} X_{i,k} X_{j,l} \\
 \text{s.t.} \quad &\sum_{v_k \in \mathcal{G}_2} X_{i,k} = 1, \forall u_i \in \mathcal{G}_1 \\
 &\sum_{u_i \in \mathcal{G}_1} X_{i,k} = 1, \forall v_k \in \mathcal{G}_2 \\
 &X_{i,k} \in \{0, 1\}, \forall u_i \in \mathcal{G}_1, v_k \in \mathcal{G}_2
 \end{aligned} \tag{1}$$

where $X \in \{0, 1\}^{|\mathcal{V}_1| \times |\mathcal{V}_2|}$ is a binary matrix representing the node matching between \mathcal{G}_1 and \mathcal{G}_2 . The value $X_{i,k}$ is 1 if node u_i in \mathcal{G}_1 matches with node v_k in \mathcal{G}_2 . The edit cost $c_{i,k}$ denotes the cost of matching u_i in \mathcal{G}_1 and v_k in \mathcal{G}_2 . $c_{i,k} = 1$ if u_i and v_k have different labels and 0 otherwise. Similarly, $c_{i,k,j,l}$ is the edit cost of matching the edge (u_i, u_j) in \mathcal{G}_1 and the edge (v_k, v_l) in \mathcal{G}_2 . $c_{i,k,j,l} = 1$ if (u_i, u_j) and (v_k, v_l) have different labels and 0 otherwise.

4 THE PROPOSED MODEL: MATA*

Distinct from previous works that formulate GED computation as a regression task, we suggest tracing the problem back to the node matching so that the combinatorial properties (*i.e.*, structure-dominant operations and multiple optimal node matchings) in GED computation could be leveraged.

4.1 Analysis of Learning to Match Nodes.

In order to learn node matchings of GED from CBQP formulation, it requires $X_{i,k} \in \{0, 1\}$ to relax to be continuous in $[0, 1]$, where the constraints in Eq. (1) could be modeled as the quadratic infeasibility penalty [20]. This relaxation strategy endows the binary matrix X with an augmented perspective. Thus, the binary matrix X can be viewed as the confidence of node u_i in \mathcal{G}_1 matching with node v_k in \mathcal{G}_2 . In this way, we formulate the problem as the linear matching paradigm, by incorporating the graph structure information into node embedding, *i.e.*, solving the following transportation problem:

$$\min \sum_{i=1}^{|\mathcal{V}_1|} \sum_{j=1}^{|\mathcal{V}_2|} X_{ij} \|\mathbf{h}_{1i} - \mathbf{h}_{2j}\|_2 \tag{2}$$

where $\mathbf{h}_1 \in \mathbb{R}^{|\mathcal{V}_1| \times d}$ and $\mathbf{h}_2 \in \mathbb{R}^{|\mathcal{V}_2| \times d}$ are the node embeddings of \mathcal{G}_1 and \mathcal{G}_2 , respectively. Intuitively, the kernel finds the optimal way to transform one set of node embeddings \mathbf{h}_1 to the other \mathbf{h}_2 ,

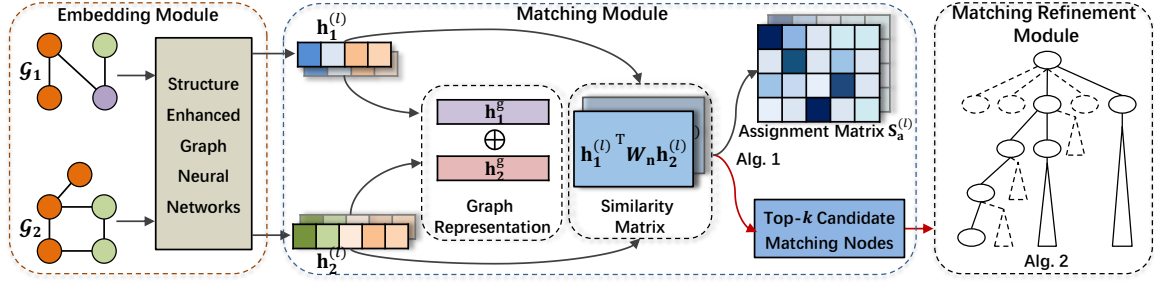


Figure 2: The framework of MATA*. The black arrows stand for the data flow in the training and testing phases and the red arrows only denote that in the testing. (1) Embedding module takes graph pairs as input and extracts the local and high-order structural information via SEGcn. (2) Matching module utilized node embeddings to build two learning tasks, i.e., learning node matching with the help of the similarity matrix and learning GED using graph representation. Further, top- k candidates and the assignment matrix are generated by Alg. 1. (3) Benefiting from the candidate nodes, MATA* only performs on the promising search directions to refine these matchings using A*LSa.

by minimizing the Euclidean distance between corresponding node pairs *w.r.t.* graph pairs.

We further analyze the combinatorial properties to design the approximate GED computation framework, after modeling it from learning node matchings by solving Eq. (2).

Structure-dominant operations. Structure operations (*node and edge insertion/deletion*) are dominant among all edit operations, which occupy at least 62.0% as illustrated in Table 1. Indeed, the operations of *node deletion* can be interpreted as *node insertion*, as we arrange the source graph \mathcal{G}_1 and target \mathcal{G}_2 graph with $\mathcal{V}_1 \leq \mathcal{V}_2$ [9]. Further, by reducing the CBQP formulation to the transportation problem in Eq. (2), the graph structure information is assumed to be embedded into node embeddings. These tell us we need a GNN to effectively learn powerful node embeddings enhanced by the graph structures (Section 4.2).

Multiple optimal node matchings. Due to the combination and permutation natures of node matchings, the two graphs typically have multiple optimal node matchings that yield the GED. Hence, directly learning the node correspondence according to the matching confidence is extremely challenging, as it could lead to the inability to satisfy the injection constraint or a larger gap between $\text{ged}(\cdot, \cdot)$ and $\text{ged}(\cdot, \cdot)$. That is, it requires us to relax the constraint on the number of matched nodes and obtain candidate nodes using a flexible parameter top- k . To conclude, we (1) need a differentiable top- k operation to enable the training for node matchings (Section 4.3) and (2) refine the matchings from the top- k candidates using A* algorithms (Section 4.4).

Thus, the proposed MATA* employs a structure-enhanced GNN (i.e., SEGcn) to learn the differentiable top- k candidate matching nodes which prunes the unpromising search directions of A*LSa [9] for approximate GED computation. The overview of MATA* is illustrated in Fig. 2 with the details of each module elaborated in the following paragraphs.

4.2 Embedding Module

As analyzed, structural information is critical to learn the fine-grained matching of node pairs for the GED computation. Here, a

structure-enhanced GNN SEGcn is proposed that jointly learns the structural information from the views of local and high-order.

Degree encoding. When matching the nodes of two graphs *w.r.t.* GED, the nodes with similar degrees are more likely to be matched. Note that, the degree is not an accurate measure of structural similarity as the edge insertions and deletions are involved. Hence, rather than directly encoding the degree with a one-hot vector, we assign each node with a learnable embedding d_i based on its degree, and the values of d_i are randomly initialized.

Position encoding. The nodes located with similar local positions are more likely to match in the GED computation. Shortest-path-distances [39], PageRank [24] and random walk [14, 22] are generally used to measure the relative position of nodes [38]. For the sake of computation efficiency, we employ the probabilities of random walks after different steps as the relative position encoding $p_i \in \mathbb{R}^t$.

$$p_i = [R_{ii}^{(1)}, R_{ii}^{(2)}, \dots, R_{ii}^{(t)}], \quad (3)$$

where $R = AD^{-1}$ is the random walk operator, t is the number of steps of random walks, and $R_{ii}^{(t)}$ refers to the landing probability of the node i to itself after the t -th step of random walk. Though the local positions are encoded via random walk, nodes with slight structure distinction may hard to be matched as the Eq. (3) is deterministic yet the editing operations from \mathcal{G}_1 to \mathcal{G}_2 do change the graph structures.

For more robust position encoding, perturbations are further injected to the original graphs to get the perturbed position encoding. Specifically, we randomly insert and remove a small portion of edges (10% utilized in experiments) to produce the perturbed graphs \mathcal{G}_{in} and \mathcal{G}_{re} , respectively. The random walk diffusion manners are further performed on \mathcal{G}_{in} and \mathcal{G}_{re} with the perturbed local positions p_i^{in} and p_i^{re} , respectively. Combined with the Eq. (3), the positional encoding is given as follow:

$$\hat{p}_i = p_i + p_i^{in} + p_i^{re}. \quad (4)$$

Local view. By concatenating (1) the node feature x_i (i.e., the attribute feature of its label), (2) degree encoding d_i , and (3) position encoding \hat{p}_i , the node embeddings with local structure $h_i^{(0)} \in \mathbb{R}^d$

are built via a multilayer perceptron (MLP):

$$h_i^{(0)} = \text{MLP}(x_i \oplus d_i \oplus \hat{p}_i), \quad \forall i \in \mathcal{V} \quad (5)$$

High-order view. We adopt the GCN [19] as the backbone of SEGNCN to learn the higher-order neighbor information. The node embeddings are aggregated from the embeddings of its adjacency nodes and itself. The l -th iteration of aggregation could be characterized as:

$$h_i^{(l)} = \sigma \left(\sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} h_j^{(l-1)} w^{(l-1)} \right) \quad (6)$$

where $h_i^{(l)} \in \mathbb{R}^d$ is the representation of node i of l -th GCN layer, \mathcal{N}_i is the set of neighbors of node i , and $w^{(l)}$ is the learned weights of l -th layer. In order to reduce the bias because of the different numbers of neighbors, the aggregated embeddings from adjacent nodes are also normalized by the total number of adjacent nodes c_{ij} . SEGNCN takes the obtained $h_i^{(0)}$ as the input embedding.

After encoding by SEGNCN, the node embeddings of \mathcal{G}_1 from the local and high-order views are denoted as $\mathbf{h}_1^{(0)} \in \mathbb{R}^{|\mathcal{V}_1| \times d}$ and $\mathbf{h}_1^{(l)} \in \mathbb{R}^{|\mathcal{V}_1| \times d}$, respectively. The node embeddings of \mathcal{G}_2 are similarly obtained.

4.3 Matching Module

The local and high-order structural affinities between two graphs has been ingeniously encoded into the node embedding space, by SEGNCN. As such, learning to match nodes is reduced to solve the Eq. (2). We thus jointly learn the matchings from both the local view and high-order view to obtain the differentiable top- k candidates by iteratively minimizing the underlying transportation problem. In addition to the task of learning node matching, a complementary task learning GED is also put forward, which attempts to learn the distance between graph representations that assists the node matching task.

Learning node matchings. Intuitively, we learn node matchings from fine-grained correspondences to minimize the transportation problem, which aims that the resultant node matchings are not just approximations but are reflective of the genuine structural alignments between the two graphs.

Similarity matrix. To solve the Eq. (2), we model it in a flexible way, and similarity matrices from the local view $\mathbf{S}^{(0)} \in |\mathcal{V}_1| \times |\mathcal{V}_2|$ and the high-order view $\mathbf{S}^{(l)} \in |\mathcal{V}_1| \times |\mathcal{V}_2|$ are:

$$\begin{aligned} \mathbf{S}^{(0)} &= \sigma(\mathbf{h}_1^{(0)\top} \mathbf{W}_n \mathbf{h}_2^{(0)}) \\ \mathbf{S}^{(l)} &= \sigma(\mathbf{h}_1^{(l)\top} \mathbf{W}_n \mathbf{h}_2^{(l)}) \end{aligned} \quad (7)$$

where $\mathbf{W}_n \in \mathbb{R}^{d \times d}$ is a learnable weights matrix and shares the parameters between $\mathbf{S}^{(0)}$ and $\mathbf{S}^{(l)}$. All elements of similarity matrix \mathbf{S} are positive after applying the sigmoid function, and $\mathbf{S}_{i,j}^{(0)}$ measures the similarity between \mathcal{V}_{1i} and \mathcal{V}_{2j} from the local view. And $\mathbf{S}_{i,j}^{(l)}$ measures the similarity from the high-order view. Besides, $\mathbf{S}^{(0)}$ also models the cost by transforming the embedding $\mathbf{h}_1^{(0)}$ to $\mathbf{h}_2^{(0)}$. Different from padding the similarity matrix [4], it is enough to represent all possible matchings with $|\mathcal{V}_1| \times |\mathcal{V}_2|$ from the Eq. 1.

Algorithm 1 Differentiable top- k matching nodes

Input: Similarity mat $\mathbf{S}^{(0)}, \mathbf{S}^{(l)}, k$, regularization ϵ

Output: Assignment mat $\mathbf{S}_a^{(0)}, \mathbf{S}_a^{(l)}$, candidates $M^{|\mathcal{V}_1| \times k}$

- 1: Build $\mathbf{D}, \mathbf{c}, \mathbf{r}$ from $\mathbf{S}^{(0)}$ by Eq. (8); $\Gamma = -\mathbf{D}/\epsilon$;
 - 2: **while** Γ is not converged **do** ▷ Sinkhorn normalization
 - 3: $\Gamma = \text{diag}((\Gamma \odot \mathbf{r}))^{-1} \Gamma$;
 - 4: $\Gamma = \text{diag}((\Gamma^\top \mathbf{1} \odot \mathbf{c}))^{-1} \Gamma$;
 - 5: Rebuild assignment mat $\mathbf{S}_a^{(0)}$ from Γ ;
 - 6: Repeat lines 1–5 for $\mathbf{S}^{(l)}$ to obtain $\mathbf{S}_a^{(l)}$;
 - 7: **if** training **then return** $\mathbf{S}_a^{(0)}, \mathbf{S}_a^{(l)}$;
 - 8: **else return** $M^{|\mathcal{V}_1| \times k}$ by greedily searching top- k ;
-

Top- k candidate matching nodes. Inspired by [31, 34], choosing the top- k matches from the similarity matrices $\mathbf{S}_{i,j}^{(0)}$ and $\mathbf{S}_{i,j}^{(l)}$ is typically formulated as an optimal transport problem, which selects the k most confident matches for each node based on the matching confidences, shown in Alg. 1.

Specifically, we first flatten the similarity matrix $\mathbf{S}^{(0)}$ with local structure affinity into $\mathbf{d} = [d_1, d_2, \dots, d_{|\mathcal{V}_1||\mathcal{V}_2|}]$. As such, to differentiable find the top- k matches, the optimal transport problem can be viewed as redistributing \mathbf{d} to one of d_{\max} and d_{\min} , where the capacities of d_{\max} and d_{\min} are k and $|\mathcal{V}_1||\mathcal{V}_2| - k$, respectively. That is the matches moved into d_{\max} are preserved during the re-distributing and the others moved into d_{\min} are discarded. Let \mathbf{c} and \mathbf{r} represent the marginal distributions, \mathbf{D} represents the distance matrix, and $\mathbf{1}$ represents the vector of all ones (line 1). And we have:

$$\begin{aligned} \mathbf{r} &= \mathbf{1}_{|\mathcal{V}_1||\mathcal{V}_2|}^\top, \quad \mathbf{c} = [|\mathcal{V}_1||\mathcal{V}_2| - k, k]^\top \\ \mathbf{D} &= \begin{bmatrix} d_1 - d_{\min} & d_2 - d_{\min} & \cdots & d_{|\mathcal{V}_1||\mathcal{V}_2|} - d_{\min} \\ d_{\max} - d_1 & d_{\max} - d_2 & \cdots & d_{\max} - d_{|\mathcal{V}_1||\mathcal{V}_2|} \end{bmatrix} \end{aligned} \quad (8)$$

Then an efficient method Sinkhorn [13, 16] for solving the optimal transport problem is typically adopted to learn the probabilities of the top- k matchings, which is an approximate and differentiable version of Hungarian. It iteratively performs row-normalization, *i.e.*, element-wise division by the sum of its row and column-normalization until convergence, where \odot means element-wise division, $\text{diag}(\cdot)$ means building a diagonal matrix from a vector (lines 2–4).

After the differentiable top- k operation, we reshape Γ into the *assignment matrix* from local view $\mathbf{S}_a^{(0)} \in |\mathcal{V}_1| \times |\mathcal{V}_2|$, which essentially measures the confidence of \mathcal{V}_{1i} and \mathcal{V}_{2j} belonging to the optimal matching (line 5). For the similarity matrix $\mathbf{S}^{(l)}$ with high-order structure affinity are also performed to obtain the *assignment matrix* $\mathbf{S}_a^{(l)}$ from high-order view (line 8). Finally, Alg. 1 returns $\mathbf{S}_a^{(0)}, \mathbf{S}_a^{(l)}$ during the training, and top- k candidate nodes $M^{|\mathcal{V}_1| \times k}$ during the testing (lines 6–7).

Note that, during the testing, we further propose a greedy method to find top- k candidate nodes in $O(kn^2)$ time. In brief, it iteratively finds a node with the largest matching probability as a candidate from the *unmatched nodes*, where the injection constraint of node matchings is also guaranteed.

Learning GED. We further propose an auxiliary task tailored to learn the (approximate) graph edit distance that assists the node matching task by exploiting the graph-level similarity.

Algorithm 2 Mapping refinement based on A*LSa [1]

Input: Graphs $\mathcal{G}_1, \mathcal{G}_2$, candidates $M^{|\mathcal{V}_1| \times k}$
Output: The approximate $\widetilde{\text{ged}}(\mathcal{G}_1, \mathcal{G}_2)$

```

1: Push  $(0, \emptyset, \text{null}, 0)$  into  $Q$ ;  $\triangleright$  Initialize the priority queue  $Q$ 
   with the root of the search tree.
2: while  $Q \neq \emptyset$  do
3:   Pop  $(i, f, pa, lb)$  with minimum  $lb$  from  $Q$ ;
4:   Compute the lower bound  $lb$  using A*LSa for each child
      $c \in M^{|\mathcal{V}_1| \times k}$  of  $f$ ;
5:   for all child  $c \in M^{|\mathcal{V}_1| \times k}$  of  $f$  do
6:     if  $i + 1 = |\mathcal{V}_1|$  then  $\widetilde{\text{ged}}(\mathcal{G}_1, \mathcal{G}_2) = c.lb$ ; break;
7:     else Push  $(i + 1, c, f, lb)$  into  $Q$ ;
8: return  $\widetilde{\text{ged}}(\mathcal{G}_1, \mathcal{G}_2)$ 

```

Intuitively, $\mathbf{h}_1^{(0)}$ and $\mathbf{h}_1^{(l)}$ capture the node features enhanced by the local and high-order graph structural information of \mathcal{G}_1 . This intricate embedding process ensures that the nodes' features are not only captured in their raw features but are also contextualized within the broader structure of the graph. Essentially, (approximate) GED measures the similarity of graph pairs from the graph level, and hence, we aggregate the node embeddings with both local and high-order views of \mathcal{G}_1 and \mathcal{G}_2 passed through MLPs for the learning GED task. And we have the following:

$$\begin{aligned}
\mathbf{h}_1^g &= \text{MLP}(\mathbf{h}_1^{(0)} \oplus \mathbf{h}_1^{(l)}) \\
\mathbf{h}_2^g &= \text{MLP}(\mathbf{h}_2^{(0)} \oplus \mathbf{h}_2^{(l)}) \\
d_{\mathcal{G}_1, \mathcal{G}_2} &= \text{MLP}(\mathbf{h}_1^g \oplus \mathbf{h}_2^g)
\end{aligned} \tag{9}$$

That is, $d_{\mathcal{G}_1, \mathcal{G}_2}$ is predicted using the MLP operation which gradually reduces the concatenated graph representations \mathbf{h}_1^g and \mathbf{h}_2^g of the graph pair. Actually, to counter this and provide a more interpretable and standardized measure, the GED are typically normalized by $\exp\{-\text{ged}(\mathcal{G}_1, \mathcal{G}_2) \times 2/(\mathcal{V}_1 + \mathcal{V}_2)\}$.

Loss design. MATA* is trained in a supervised manner for graph pairs \mathcal{G}_x and \mathcal{G}_y using normalized ground-truth GED $d_{x,y}^t$ and its corresponding node matching $\mathcal{M}_{x,y}^t$. The loss function evaluates both the difference for predicted node matchings from the local/high-order view of assignment matrices $\mathbf{S}_a^{(0)}/\mathbf{S}_a^{(l)}$ and learning GED from the predicted normalized GED $d_{x,y}$. For the learning node matching task, we jointly minimize the negative log-likelihood of the node matchings on the assignment matrices $\mathbf{S}_a^{(0)}$ and $\mathbf{S}_a^{(l)}$:

$$\mathcal{L}_n = -\frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} \sum_{(i,j) \in \mathcal{M}_{x,y}^t} \log \mathbf{S}_{a_{i,j}}^{(0)} + \log \mathbf{S}_{a_{i,j}}^{(l)} \tag{10}$$

Note that, different from the use of permutation cross-entropy loss [32] or Hungarian loss [40] for the graph matching task, only the node pairs belonging to a node matching are penalized by \mathcal{L}_n , the other node pairs are not penalized. The rationale behind this is that multiple optimal node matchings typically exist, and these unmatched node pairs may also belong to other node matchings corresponding to the GED.

For the learning GED task, we minimize the MSE loss:

$$\mathcal{L}_g = \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} (d_{x,y} - d_{x,y}^t)^2 \tag{11}$$

Table 2: Statistics of datasets. The graph pairs are partitioned 60%, 20%, 20% as training, validation, test sets, respectively.

	Graphs	Pairs	avg($\frac{ \mathcal{E} }{ \mathcal{V} }$)	min($ \mathcal{V} $)	max($ \mathcal{V} $)	avg($ \mathcal{V} $)
AIDS	700	490K	0.98	2	10	8.90
IMDB	1500	2.25M	4.05	7	89	13.00
CANCER	800	100K	1.08	21	90	30.79

where \mathcal{D} is the set of training graph pairs.

Our final loss function is a combination of the negative log-likelihood loss and MSE loss: $\mathcal{L} = \mathcal{L}_g + \mathcal{L}_n$.

4.4 Mapping Refinement Module

MATA* finally integrates A*LSa algorithm [9] to refine the edit distance (*i.e.*, node matching) among the learned top- k candidate matching nodes $M^{|\mathcal{V}_1| \times k}$, as shown in Alg. 2.

Specifically, MATA* conducts a best-first search by treating GED computation as a pathfinding problem. Such a representation is convenient because it provides a systematic and heuristic way to explore possible node matchings. To facilitate this search process, a priority queue is maintained to store the search states during the process for guiding the search direction. The priority queue Q contains the level i , current partial matching f , the parent matching pa , and the lower bound lb . A*LSa initializes the priority queue Q by the root of the search tree (line 1). It then iteratively pops (i, f, pa, lb) from Q with the minimum lower bound, and subsequently extends the current matching f by computing the lower bound of each child belonging to the candidates $M^{|\mathcal{V}_1| \times k}$ (lines 2–7). If the full node matching is formed, then $\widetilde{\text{ged}}(\cdot, \cdot)$ equals its lower bound and is returned (lines 6, 8).

Note that, different from the hybrid approach [33, 37], during the mapping refinement of MATA*, the search space is pruned by the theoretical bounded estimation of unmatched subgraphs of A*LSa.

5 EXPERIMENTS

5.1 Experimental Settings

Datasets. In this work, three benchmark datasets *i.e.*, AIDS [3], IMDB [36], and CANCER¹ are employed. (1) AIDS is a set of antivirus screen chemical compounds labeled with 29 types. Following [3, 33], 700 graphs with no more than ten nodes are sampled as the AIDS dataset. (2) IMDB consists of 1,500 ego-networks of movie actors or actresses and each of which is a non-attributed graph. (3) CANCER consists of 32,577 graphs of molecules discovered in carcinogenic tumors. To test the scalability and efficiency of our MATA*, we sample 800 graphs with nodes from 21 to 90 as CANCER dataset, where the nodes are labeled with 37 types of atoms. Statistics of the three real-life datasets are shown in Table 2.

Baseline methods. Our baselines include three types of methods, combinatorial search-based algorithms, learning-based models and hybrid approaches. (1) The representative methods in the first category include three well-known approximate algorithms A*Beam [25], Hungarian [28] and VJ [15]. (2) The second category includes three common-used and one state-of-the-art learning models, *i.e.*, SimGNN [3], GMN [23], GENN [33] and GREED [27]. (3)

¹<https://cactus.nci.nih.gov/download/nci/CAN2DA99.sdz>

Table 3: Effectiveness evaluations. The metrics are calculated on the normalized edit distance. \uparrow indicates the high the better and \downarrow otherwise. Top- k are set to 4, 6 and 8 for AIDS, IMDB and CANCER, respectively. The unit of metrics ACC and MSE are % and 10^{-2} , respectively, and – refers to memory overflow on 32GB machines or runs in more than 10 minutes for one graph pair.

Datasets	Methods	Edit Path	ACC \uparrow	MAE \downarrow	MSE \downarrow	p@10 \uparrow	p@20 \uparrow	ρ \uparrow	τ \uparrow
AIDS	A*Beam [25]	✓	16.68	0.092	1.37	0.460	0.470	0.720	0.546
	Hungarian [28]	✓	4.19	0.194	4.77	0.293	0.328	0.541	0.386
	VJ [15]	✓	0.95	0.216	5.64	0.215	0.273	0.543	0.387
	SimGNN [3]	×	0.01	0.036	0.22	0.470	0.540	0.886	0.725
	GMN [23]	×	0.02	0.034	0.19	0.401	0.489	0.750	0.673
	GREED [27]	×	0.00	0.031	0.17	0.461	0.533	0.894	0.732
	GENN [33]	×	0.02	0.031	0.17	0.441	0.525	0.898	0.738
	GENNA* [33]	✓	20.05	0.034	0.46	0.407	0.556	0.515	0.378
	MATA* (Ours)	✓	59.12	0.029	0.37	0.486	0.526	0.844	0.698
IMDB	A*Beam [25]	✓	23.18	0.111	5.22	0.464	0.527	0.489	0.381
	Hungarian [28]	✓	22.53	0.115	5.38	0.438	0.498	0.465	0.359
	VJ [15]	✓	22.24	0.115	5.38	0.436	0.495	0.465	0.359
	SimGNN [3]	×	0.11	0.114	5.01	0.474	0.531	0.500	0.388
	GMN [23]	×	0.29	0.128	5.01	0.479	0.542	0.513	0.392
	GREED [27]	×	0.93	0.110	5.04	0.477	0.541	0.499	0.389
	GENN [33]	×	0.22	0.108	5.04	0.476	0.533	0.495	0.384
	GENNA* [33]	✓	–	–	–	–	–	–	–
	MATA* (Ours)	✓	44.80	0.098	5.03	0.509	0.570	0.542	0.456
CANCER	A*Beam [25]	✓	44.23	0.053	1.14	0.161	0.266	0.446	0.352
	Hungarian [28]	✓	2.19	0.162	3.56	0.123	0.227	0.139	0.096
	VJ [15]	✓	0.00	0.184	4.85	0.095	0.187	0.188	0.133
	SimGNN [3]	×	0.01	0.068	1.42	0.273	0.297	0.277	0.191
	GMN [23]	×	0.00	0.071	1.47	0.280	0.285	0.254	0.174
	GREED [27]	×	0.00	0.077	1.86	0.131	0.164	0.170	0.118
	GENN [33]	×	0.00	0.069	1.44	0.285	0.264	0.300	0.207
	GENNA* [33]	✓	–	–	–	–	–	–	–
	MATA* (Ours)	✓	55.89	0.040	1.13	0.820	0.825	0.729	0.625

We chose an up-to-date model GENNA* as the representative of hybrid approaches, and our MATA* also belongs to this category.

Evaluation metrics. We adopt the following experimental metrics to evaluate the performance of the various approaches: (1) Edit path means whether a method can recover the edit path corresponding to the computed edit distance. (2) Accuracy (ACC), which measures the accuracy between the computed distance and the ground-truth solutions. (3) Mean Absolute Error (MAE), which indicates the average discrepancy between the computed distance and ground truth. (4) Mean Squared Error (MSE), which stands for the average squared difference between the computed distance and ground truth. (5) Precision at 10 (p@10) and (6) Precision at 20 (p@20), both of which mean the precision of the top 10 and 20 most similar graphs within the ground truth top 10 and 20 similar results. (7) Spearman’s Rank Correlation Coefficient (ρ) and (8) Kendall’s Rank

Correlation (τ), both of which measure how well the computed results match with the ground-truth ranking results. (9) Time. It records the running time to compute the distance for one graph pair. The methods involving learning only report the testing time.

Due to the fact that exact GED computation is NP-complete, the ground-truth of AIDS is produced by exact algorithms and the ground-truth of IMDB and CANCER are generated by the smallest edit distances of A*Beam, Hungarian, and VJ, following [3]. Note that, MATA* is able to achieve a smaller edit distance, and the ground-truth of IMDB and CANCER are further updated by the best results of the four approaches. Therefore, the metrics on AIDS are calculated by the exact solutions and the metrics on IMDB and CANCER are calculated by the updated ground-truth. Note that the edit distance is normalized into a similarity score in the range of (0, 1] as explained in Section 4.3, the same as [3, 33].

Table 4: Efficiency evaluations. Average running time for solving one graph pair on test data (ms). The training time for learning-based and hybrid approaches does not include.

	SimGNN	GMN	GENN	A*Beam	Hungarian	VJ	GENNA*	MATA*
AIDS	0.3	9.0	0.4	20.4	6.7	6.7	38624	4.4
IMDB	0.7	5.9	0.4	26.6	230.8	230.7	–	35.3
CANCER	5.5	91.5	9.5	271.7	38.8	32.7	–	146.8

Parameter settings. We conduct all experiments on machines with Intel Xeon Gold@2.40GHz CPU and NVIDIA Tesla V100 32GB GPU. The number of SEGcN layers, *i.e.*, l is set to 3 and the random walk step t is set to 16 for the three datasets. During the training, we set the batch size to 128 and use Adam optimizer with 0.001 learning rate and 5×10^4 weight decay for each dataset. The source codes and data are available at <https://github.com/jfkey/mata>.

5.2 Experimental Results

In this section, we evaluate the performance of MATA* from the effectiveness, scalability, efficiency, ablation study, top- k comparisons, and the analysis of assignment matrices.

Effectiveness evaluations. Table 3 shows the effectiveness of eight approaches on three real-world datasets. MATA* consistently achieves the best performance under almost each evaluation metric, which demonstrates the superiority of our hybrid method MATA* incorporating the two combinatorial properties of GED computation. We conduct the following findings from the evaluations.

(1) From the ACC, MATA* achieves smaller edit distances at least (58.1%, 32.1%, 53.6%) of graph pairs on (AIDS, IMDB, CANCER) when comparing with combinatorial search-based and hybrid approaches. Hence, the ground-truth of IMDB and CANCER are further updated by these, which reduces MAE by at least (6.5%, 9.1%, 24.5%). (2) Only learning-based models cannot recover the edit path, as they directly learn GED as a similarity score and ignore the combinatorial nature. (3) On IMDB, all methods perform worse than on AIDS and CANCER. IMDB is large with a wide range from 7 to 89 nodes. Besides, the graphs are much denser with $|\mathcal{E}|/|\mathcal{V}| = 4.05$ and the distances of pairs are also larger, which increases the difficulty for combinatorial search and learning methods.

(4) The improvement of MATA* on ACC is such significant with at least (39.0%, 21.6%, 11.7%), and the improvement of other metrics is relatively less significant. The rationales behind this lie in that (a) MATA* models from the perspective of node matching and explicitly build the task of learning node matching, that is the learned top- k candidate nodes could directly improve the accuracy due to the correspondence between GED and the node matching. (b) For fewer node pairs whose matchings have not been learned by MATA*, it prunes the search subtree rooted at the these node pairs, which leads to a larger edit distance reflected in other metrics.

Scalability w.r.t. graph size. Consider the overall performance of the approaches from small-size, *i.e.*, AIDS to large-size *i.e.*, IMDB, CANCER. We find the following. (1) Our MATA* leverages the learned candidate matching nodes to directly prune unpromising search directions, which scales well to large-size graphs and also performs better on *i.e.*, IMDB and CANCER from Table 3. (2) Combinatorial

Table 5: Ablation study. / SEGcN refers to replacing SEGcN with GCN, / LN refers to only removing learning node matching, / LG refers to only removing learning GED, and / A* refers to only removing the matching refinement module.

Datasets	Models	ACC \uparrow	MAE \downarrow	p@10 \uparrow	ρ \uparrow
AIDS	MATA*	59.12	0.031	0.486	0.844
	/ SEGcN	54.38	0.036	0.485	0.819
	/ LN	32.13	0.057	0.397	0.783
	/ LG	58.09	0.033	0.469	0.858
	/ A*	9.28	0.167	0.244	0.357
IMDB	MATA*	44.80	0.098	0.509	0.542
	/ SEGcN	41.43	0.100	0.493	0.544
	/ LN	40.01	0.102	0.488	0.541
	/ LG	42.99	0.098	0.505	0.549
	/ A*	30.16	0.112	0.428	0.521
CANCER	MATA*	55.89	0.040	0.820	0.729
	/ SEGcN	53.36	0.042	0.817	0.699
	/ LN	45.85	0.046	0.790	0.582
	/ LG	55.87	0.040	0.812	0.729
	/ A*	10.89	0.104	0.327	0.146

search-based algorithms can be extended to large-scale graphs with general performance due to aggressive relaxation (Hungarian and VJ) or pruning strategies (A*Beam). (3) Learning-based models add a bias to the predicted GED values to reduce the discrepancy between the predicted and ground truth. Their scalability heavily relies on the ground truth produced by combinatorial search-based algorithms. This is why they perform worse on IMDB and CANCER than AIDS. (4) The hybrid approach GENNA* only completes AIDS for less than 10 nodes graphs and fails to scale to IMDB and CANCER. This is because GENNA* explores the entire space, *i.e.*, factorial scale and takes $O(n^2d + d^2n)$ time for each search as explained in the introduction.

Efficiency w.r.t. running time. The computational efficiency of eight approaches, evaluated over three real-life datasets, has been presented in Table 4. Due to the end-to-end learning, SimGNN and GENN achieve the best results and run in several microseconds to predict the GED for one graph pair. Though MATA* is slightly slower than the learning-based models, its running time is close to combinatorial search-based algorithms, and nearly 10^4 times faster than the other hybrid approach GENNA*. This marked difference in performance underscores the significance of top- k candidate node finds and the impact on the computational efficiency of MATA*.

Ablation study. We perform ablation studies to verify the effectiveness of each module and the results are reported in Table 5. It is observed the performance drops dramatically if the matching refinement module is removed. This is because multiple optimal node matchings typically exist from the node matching perspective of GED. It is also noted that the performance considerably decreases if the structure-enhanced GNN (*i.e.*, SEGcN) is replaced by the vanilla one. It demonstrates that SEGcN successfully captures the combinatorial property of structure-dominant operations and learns the

Table 6: Performance evaluations w.r.t. different top- k . The metrics are calculated in the same way as those in Table 3.

Datasets	top- k	ACC \uparrow	MAE \downarrow	p@10 \uparrow	ρ \uparrow	Time \downarrow
AIDS	5	74.85	0.018	0.586	0.796	4.3
	6	84.10	0.010	0.693	0.859	4.4
	7	91.79	0.005	0.787	0.910	4.4
	8	95.79	0.002	0.846	0.940	4.5
	9	97.76	0.001	0.912	0.959	4.8
	10	100.00	0.000	1.000	1.000	5.2
IMDB	5	38.57	0.106	0.381	0.578	30.2
	6	39.40	0.105	0.391	0.579	35.3
	7	40.97	0.103	0.401	0.582	43.3
	8	41.56	0.102	0.400	0.582	51.9
	9	44.47	0.100	0.412	0.587	53.2
	10	45.05	0.100	0.415	0.588	55.3
CANCER	5	5.01	0.104	0.452	0.678	129.8
	6	7.37	0.091	0.486	0.709	146.8
	7	10.55	0.079	0.543	0.734	153.1
	8	14.37	0.069	0.569	0.758	168.0
	9	22.63	0.058	0.615	0.777	176.7
	10	34.19	0.048	0.684	0.794	193.2

powerful embeddings for approximate GED computation. Further, the two designed learning tasks (*i.e.*, LN, and LG) are both helpful for improving the solution quality of GED, especially the learning node matching tasks (*i.e.*, LN). The ablation study is consistent with our analysis of the approximate GED computation.

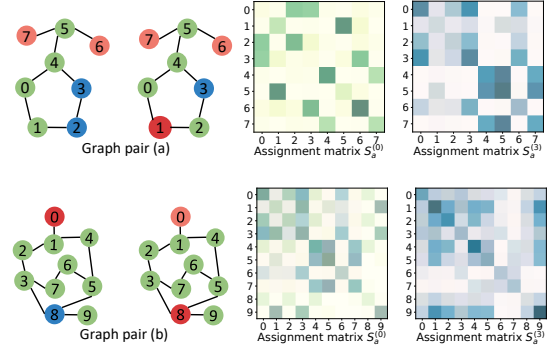
5.3 Performance w.r.t. top- k selection.

We also study the performance w.r.t. selecting different k of MATA* on AIDS, IMDB and CANCER datasets. We conduct the following findings from Table 6.

(1) Varying k in the experiments emphasize the trade-off between solution quality and time. That is setting larger k could improve the approximate GED solution quality, but the running time indeed increases mainly due to the large search space of A*LSa. (2) AIDS is a small dataset with the exact solutions, and MATA* also achieves the optimal solutions running in 5.2 ms, when k is set to 10. Note that, MATA* degenerates to A*LSa when all nodes of \mathcal{G}_2 are selected as the candidate matching nodes. (3) When k is set to 6, 8 for datasets IMDB and CANCER, the evaluation metric is worse than that in Table 3. In fact, k is set to 10 in this test which achieves a smaller edit distance, and the ground-truth of IMDB and CANCER are further updated by these solutions. Hence, the evaluation metrics are recalculated, which produces worse metrics on IMDB and CANCER. To encapsulate, the interplay of the value of k , offers a balance between computational scalability and accuracy of the approximate graph edit distance computation.

5.4 Analysis of assignment matrices of MATA*

we offer a visual representation of four assignment matrices that encapsulate both local and high-order perspectives. These matrices,

**Figure 3: Analysis of the assignment matrices. The GED of graph pairs (a) & (b) are both equal to 2.**

specifically pertaining to two pairs of graphs, have been portrayed as heatmap images, and are generated by MATA* on AIDS in Fig. 3. From our observations, both the local and high-order views play a crucial role in the extraction of features tailored for node matchings. This is evident when considering specific node pairs, such as (6,6) and (7,7), from the graph pair labeled (a) (the top row of the assignment matrices $S_a^{(0)}$ and $S_a^{(l)}$). We can see that the local and high-order views both extract features for matching nodes, *e.g.*, the node pairs (6,6) and (7,7) of the pair (a) (the top row). Besides, the assignment matrix $S_a^{(l)}$ in the high-order view typically has a more powerful capacity to learn the node correspondence compared to the local view $S_a^{(0)}$. For example, $S_a^{(0)}$ fails to capture the node pairs (1,1) of the pair (b), while $S_a^{(l)}$ successfully learns it. Thus, Fig. 3 shows the importance of using local and high-order views to jointly learn the top- k candidates rather than a single one.

6 CONCLUSION

We have presented a data-driven hybrid approach MATA* based on Graph Neural Networks (SEGNN) and A* algorithms, which leverage the learned candidate matching nodes to prune unpromising search directions of A*LSa algorithm to approximate graph edit distance. We have modeled it from a new perspective of node matching and combined the intrinsic relationship between GED computation and node matching. Besides, the design of our hybrid approach MATA* has been aware of the two combinatorial properties involved in GED computation: structure-dominant operations and multiple optimal node matching, to learn the matching nodes from both local and high-order views. Benefiting from the candidate nodes, MATA* has offered a balance between computational scalability and accuracy on the real-life datasets. Finally, we have conducted extensive experiments on AIDS, IMDB, and CANCER to demonstrate the effectiveness, scalability, and efficiency of combinatorial search-based, learning-based and hybrid approaches.

ACKNOWLEDGMENTS

This work is supported in part by NSF of China under Grant 61925203 & U22B2021. For any correspondence, please refer to Shuai Ma and Min Zhou.

REFERENCES

- [1] Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. 2015. An Exact Graph Edit Distance Algorithm for Solving Pattern Recognition Problems. In *ICPRAM*. 271–278.
- [2] Jiyang Bai and Peixiang Zhao. 2021. TaGSim: Type-aware Graph Similarity Learning and Computation. *Proc. VLDB Endow.* 15, 2 (2021), 335–347.
- [3] 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 *WSDM*. 384–392.
- [4] 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 *AAAI*. 3219–3226.
- [5] David B. Blumenthal, Nicolas Boria, Johann Gamper, Sébastien Bougleux, and Luc Brun. 2020. Comparing heuristics for graph edit distance computation. *VLDB J.* 29, 1 (2020), 419–458.
- [6] David B. Blumenthal and Johann Gamper. 2017. Exact Computation of Graph Edit Distance for Uniform and Non-uniform Metric Edit Costs. In *GbRPR*, Vol. 10310. 211–221.
- [7] David B. Blumenthal and Johann Gamper. 2020. On the exact computation of the graph edit distance. *Pattern Recognit. Lett.* 134 (2020), 46–57.
- [8] Garcia-Hernandez Carlos, Alberto Fernández, and Francesc Serratos. 2019. Ligand-Based Virtual Screening Using Graph Edit Distance as Molecular Similarity Measure. *J. Chem. Inf. Model.* 59, 4 (2019), 1410–1421.
- [9] Lijun Chang, Xing Feng, Xuemin Lin, Lu Qin, Wenjie Zhang, and Dian Ouyang. 2020. Speeding Up GED Verification for Graph Similarity Search. In *ICDE*. 793–804.
- [10] Lijun Chang, Xing Feng, Kai Yao, Lu Qin, and Wenjie Zhang. 2023. Accelerating Graph Similarity Search via Efficient GED Computation. *IEEE Trans. Knowl. Data Eng.* 35, 5 (2023), 4485–4498.
- [11] Xiaoyang Chen, Hongwei Huo, Jun Huan, and Jeffrey Scott Vitter. 2019. An efficient algorithm for graph edit distance computation. *Knowl. Based Syst.* 163 (2019), 762–775.
- [12] Minsu Cho, Karteek Alahari, and Jean Ponce. 2013. Learning graphs to match. In *ICCV*. 25–32.
- [13] Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *NeurIPS* 26 (2013), 2292–2300.
- [14] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2022. Graph Neural Networks with Learnable Structural and Positional Representations. In *International Conference on Learning Representations*.
- [15] Stefan Fankhauser, Kaspar Riesen, and Horst Bunke. 2011. Speeding Up Graph Edit Distance Computation through Fast Bipartite Matching. In *GbRPR*, Vol. 6658. 102–111.
- [16] Matthias Fey, Jan Eric Lenssen, Christopher Morris, Jonathan Masci, and Nils M. Kriege. 2020. Deep Graph Matching Consensus. In *International Conference on Learning Representations*.
- [17] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* 4, 2 (1968), 100–107.
- [18] Jongik Kim, Dong-Hoon Choi, and Chen Li. 2019. Inves: Incremental Partitioning-Based Verification for Graph Similarity Search. In *EDBT*. 229–240.
- [19] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR* abs/1609.02907 (2016).
- [20] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. 2014. The unconstrained binary quadratic programming problem: a survey. *Journal of combinatorial optimization* 28 (2014), 58–81.
- [21] Danaï Koutra, Hanghang Tong, and David Lubensky. 2013. Big-align: Fast bipartite graph alignment. In *ICDM*. 389–398.
- [22] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance encoding: Design provably more powerful neural networks for graph representation learning. *NeurIPS* 33 (2020), 4465–4478.
- [23] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. 2019. Graph Matching Networks for Learning the Similarity of Graph Structured Objects. In *ICML*, Vol. 97. 3835–3845.
- [24] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. 2021. GraphiT: Encoding Graph Structure in Transformers. *CoRR* abs/2106.05667 (2021).
- [25] Michel Neuhaus, Kaspar Riesen, and Horst Bunke. 2006. Fast Suboptimal Algorithms for the Computation of Graph Edit Distance. In *IAPR Workshops*, Vol. 4109. 163–172.
- [26] Yun Peng, Byron Choi, and Jianliang Xu. 2021. Graph Edit Distance Learning via Modeling Optimum Matchings with Constraints. In *IJCAI*. 1534–1540.
- [27] Rishabh Ranjan, Siddharth Grover, Sourav Medya, Venkatesan Chakaravarthy, Yogish Sabharwal, and Sayan Ranu. 2022. Greed: A neural framework for learning graph distance functions. *NeurIPS* 35 (2022), 22518–22530.
- [28] Kaspar Riesen and Horst Bunke. 2009. Approximate graph edit distance computation by means of bipartite graph matching. *Image Vis. Comput.* 27, 7 (2009), 950–959.
- [29] Kaspar Riesen, Sandro Emmenegger, and Horst Bunke. 2013. A Novel Software Toolkit for Graph Edit Distance Computation. In *GbRPR*, Vol. 7877. 142–151.
- [30] Kaspar Riesen, Stefan Fankhauser, and Horst Bunke. 2007. Speeding Up Graph Edit Distance Computation with a Bipartite Heuristic. In *MLG*. 21–24.
- [31] Runzhong Wang, Ziao Guo, Shaofei Jiang, Xiaokang Yang, and Junchi Yan. 2023. Deep Learning of Partial Graph Matching via Differentiable Top-K. In *CVPR*. 6272–6281.
- [32] Runzhong Wang, Junchi Yan, and Xiaokang Yang. 2019. Learning Combinatorial Embedding Networks for Deep Graph Matching. In *ICCV*. 3056–3065.
- [33] Runzhong Wang, Tianqi Zhang, Tianshu Yu, Junchi Yan, and Xiaokang Yang. 2021. Combinatorial Learning of Graph Edit Distance via Dynamic Embedding. In *CVPR*. 5241–5250.
- [34] Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. 2020. Differentiable top-k with optimal transport. *NeurIPS* 33 (2020), 20520–20531.
- [35] Junchi Yan, Shuang Yang, and Edwin R. Hancock. 2020. Learning for Graph Matching and Related Combinatorial Optimization Problems. In *IJCAI*. 4988–4996.
- [36] Pinar Yanardag and S. V. N. Vishwanathan. 2015. Deep Graph Kernels. In *SIGKDD*. 1365–1374.
- [37] Lei Yang and Lei Zou. 2021. Noah: Neural-optimized A* Search Algorithm for Graph Edit Distance Computation. In *ICDE*. 576–587.
- [38] Menglin Yang, Min Zhou, Lujia Pan, and Irwin King. 2023. κHGCN: Tree-likeness Modeling via Continuous and Discrete Curvature Learning. In *SIGKDD*. 2965–2977.
- [39] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? *NeurIPS* 34 (2021), 28877–28888.
- [40] Tianshu Yu, Runzhong Wang, Junchi Yan, and Baoxin Li. 2020. Learning deep graph matching with channel-independent embedding and Hungarian attention. In *International Conference on Learning Representations*.
- [41] Zhen Zhang, Jiajun Bu, Martin Ester, Zhao Li, Chengwei Yao, Zhi Yu, and Can Wang. 2021. H2MN: Graph Similarity Learning with Hierarchical Hypergraph Matching Networks. In *SIGKDD*. 2274–2284.
- [42] Wei Zhuo and Guang Tan. 2022. Efficient Graph Similarity Computation with Alignment Regularization. *NeurIPS* 35, 30181–30193.