MATA*: COMBINING LEARNABLE NODE MATCHING WITH A* ALGORITHM FOR APPROXIMATE GRAPH EDIT DISTANCE COMPUTATION

Anonymous authors
Paper under double-blind review

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 and A* algorithms, which models from the perspective of learning to match nodes instead of directly regressing GED. That is it leverages the learned node matchings to prune unpromising search directions of the A* algorithm. Specifically, aware of the combinatorial property of structure-dominant operations (*i.e.,* node and edge insertion/deletion) in GED computation, a structure-enhanced Graph Neural Network is firstly designed to effectively learn powerful node embeddings *w.r.t.* node matchings. Based on this, the pairwise node similarity matrix is next built. Second, top-k candidate matching nodes are produced from the similarity matrix which is adhering to the combinatorial property of 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 demonstrate the superiority of MATA* as it significantly outperforms the combinatorial search-based, learning-based and hybrid approaches and scales well to large-size graphs.

1 INTRODUCTION

Graphs are ubiquitous and widely used for structured data modeling in many domains, such as chemical compounds Carlos et al. (2019), social networks Fey et al. (2020), compute vision Yan et al. (2020) and programming languages Li et al. (2019). One of the fundamental issues related to graph-based applications is to compute the graph similarity, among which graph edit distance (GED) is a widely used metric due to its flexible and domain-agnostic features Li et al. (2019); Chang et al. (2020; 2022); Bai et al. (2019). In general, GED computation refers to finding the minimum number of edit operations (*node insertion/deletion, edge insertion/deletion, and node/edge relabeling*) to transform the source graph to a target one Blumenthal et al. (2020) (see Fig. 1 for an example).

Exact GED computation guarantees optimality which is however NP-complete Kim et al. (2019); Chen et al. (2019); Chang et al. (2020). It typically treats all possible edit operations as a pathfinding problem where A* algorithm (a best-first search) is widely used to expand the search. 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 Blumenthal & Gamper (2020).

Some recent works for approximate GED computation have been proposed with the help of the graph representation techniques, which can be divided into two main categories: Learning-based

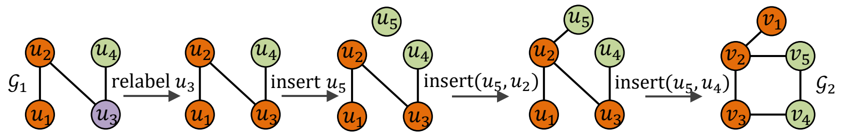

Figure 1: An edit path from source graph $\mathcal{G}_1$ to target graph $\mathcal{G}_2$. Different colors represent the node with different labels. Indeed, ged $(\mathcal{G}_1, \mathcal{G}_2) = 4$, *i.e.,* at least four edit operations are required to transform $\mathcal{G}_1$ to $\mathcal{G}_2$ and 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 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*.

models Li et al. (2019); Bai et al. (2019); Bai & Zhao (2021); Bai et al. (2020); Peng et al. (2021) and hybrid approaches Wang et al. (2021); Yang & Zou (2021). (1) For learning-based models, they directly formulate the approximate GED computation as a regression task and supervised learn the GED as a similarity score in an end-to-end manner. Although such learning-based methods can speed up approximate computation, they could encounter the *inaccurate GED approximation* issue (*i.e.,* the predicted GED is smaller than the exact) and also fail to recover an actual edit path, which is indispensable in specific tasks *e.g.,* network alignment Koutra et al. (2013), graph matching Cho et al. (2013); Wang et al. (2021). (2) For hybrid approaches, recently Wang et al. (2021) and Yang & Zou (2021) separately propose two hybrid approaches, both of which apply Graph Neural Networks (GNNs) to optimize the search directions of A* algorithms. However, even though they exhaustively explore the search space, they still fail to find the optimal due to *inaccurate GED approximation* in the cost function estimation (*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 $\mathcal{O}(n^2d+d^2n)$ time for extending each search, and encounter scalability issues Wang et al. (2021).

It is known that GED computation equals finding the optimal node matching between the source and the target graphs. Once the node matching is given, then GED can be easily calculated by only scanning the two graphs once Chang et al. (2020), 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 of 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 directly 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 modeling in end-to-end learning. (2) *Structure-dominant operations* (*i.e.,* most edit operations are involved in structure) create challenges to incorporate 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 **mat**ching nodes to prune unpromising search directions of the up-to-date **A*** algorithm (*i.e.,* A*LSa Chang et al. (2020)) for approximate GED computation.

Contributions. Our main contributions are summarized 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.,* SEGCN) is put forward to learn powerful 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 matching nodes are produced to be aware of the multiple optimal node matchings combinatorial property, which is built upon two complementary learning tasks, *i.e.,* learning GED and learning node matching.

(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 (45.15%,

21.54%, 11.35%) and reduces the average discrepancy by (12.4%, 9.1%, 24.5%) at least on (AIDS, IMDB, CANCER), respectively.

2 RELATED WORKS

Computing the graph edit distance between graphs is a classical combinatorial optimization problem over graphs and an extensive body of literature exists in various domains. (1) Combinatorial search-based includes the exact methods by exhaustive searching the entire space Riesen et al. (2007); Chang et al. (2020; 2022); Kim et al. (2019) and the approximate methods by trading-off sub-optimal and efficiency Neuhaus et al. (2006); Riesen & Bunke (2009); Fankhauser et al. (2011). (2) Learning-based models use graph representation techniques to learn the GED as a similarity of graphs Bai et al. (2019); Li et al. (2019); Peng et al. (2021); Bai & Zhao (2021). (3) Recently, hybrid approaches have been proposed which learn good heuristics from data to guide the search for combinatorial algorithms Wang et al. (2021); Yang & Zou (2021). In Appendix A, we present a detailed overview of existing literature and highlight the connections and differences between our approach and related studies.

3 PRELIMINARIES

We focus the discussions on the labeled and undirected simple graphs, that is a graph is 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 assign 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 number 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 $\mathsf{ged}(\mathcal{G}_1, \mathcal{G}_2)$ Bai & Zhao (2021).

Proposition 1: *Given graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ with $|\mathcal{V}_1| \leq |\mathcal{V}_2|$, there is no node deletion in optimal edit operations to transform $\mathcal{G}_1$ to $\mathcal{G}_2$ Chang et al. (2020).* □

Based on the commutativity of $\mathsf{ged}(\cdot, \cdot)$ and Proposition 1, *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.

Proposition 2: *The* ged *between $\mathcal{G}_1$ and $\mathcal{G}_2$ equals the minimum edit cost among all node matchings from $\mathcal{G}_1$ to $\mathcal{G}_2$ Chang et al. (2020).* □

Proposition 2 reveals the equivalence of the GED computation and the optimal node matchings. Here, the node matching refers to an injective function from $\mathcal{V}_1$ to $\mathcal{V}_2$ as $|\mathcal{V}_1| \leq |\mathcal{V}_2|$.

Note that, GED describes the similarity of two graphs by edit operations where finding candidate optimal node matchings is the core of GED computation and the nodes with similar attributes and local structures are more likely to be matched Chang et al. (2020; 2022); Fey et al. (2020). Propositions 1 & 2 provide GED computation from the node matching perspective, and our hybrid approach essentially learns to match nodes, which is inspired by these.

4 THE PROPOSED MODEL: MATA*

Different from formulating GED computation as a regression task via end-to-end learning, we model it from the perspective of node matching and further incorporate the two combinatorial properties (*i.e.,*structure-dominant operations and multiple optimal node matchings) to design the hybrid approach MATA*. It combines a structure-enhanced GNN (*i.e.,* SEGCN) and an up-to-date combinatorial search-based algorithm A*LSa for approximate GED computation.

Our approach MATA* consists of three components: embedding module, matching module, and optimal finding module. The overview of MATA* is illustrated in Fig. 2.

4.1 EMBEDDING MODULE

From learning matching nodes to model GED computation, graph structural information is important for graph edit distance computation task. (1) Structure operations (*node and edge insertion/dele-*

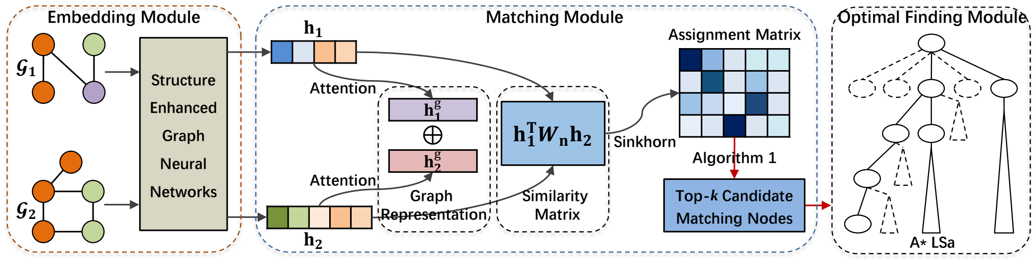

Figure 2: The framework of MATA*. The black arrows stand for the data flow in the training and testing phases and the red arrows only denote that in the testing. (1) Embedding module takes graph pairs as input and the fine-grained structural information is encoded into node embeddings via a GNN tailored to the GED computation: SEGCN. (2) Matching module using node embeddings to build two learning tasks, *i.e.,* learning GED using graph representation and learning node matching with the help of the pairwise node similarity matrix. Further, top-k candidate matching nodes are generated from the assignment matrix. (3) Benefiting from the candidate nodes, MATA* only performs on the promising search directions to find the optimal among these using A*LSa.

tion) lie the core of edit operations, and structure-dominant operations are a vital combinatorial property. As illustrated in Table 1, for AIDS and CANCER, more than 62.0% of operations belong to structure operations, while all of IMDB belong to structure operations as they are non-attribute graphs. Besides, (2) GED intrinsically describes the similarity between graphs by the number of edit operations, preference is given to matching nodes with similar attributes and local structures, from the heuristic Fey et al. (2020).

Actually, there are many ways to encode the graph structural information such as Ying et al. (2021); Mialon et al. (2021); Dwivedi et al. (2022), and we encode the structural information relevant to the GED task into our SEGCN in a simple and low-complexity way, which is described as follows.

Node feature encoding. We first initialize node features (x_i) for each node. Specifically, for attribute graphs, *e.g.,* chemical compound graphs, we encode each node to a one-hot feature based on its label. For non-attribute graphs, *e.g.,* actor/actress ego-networks, we encode all nodes with the same weights as initialized node features.

Importance encoding. We next propose the importance encoding (c_i) to capture the importance of different nodes. The node becomes important when it closely interacts with its local neighborhoods, and such information has a different impact when building the similarity between nodes. For example, the node u_1 in $\mathcal{G}_1$ with a degree of 0 is more likely to match the node v_1 also with a degree of 0 in $\mathcal{G}_2$ in Fig. 1. To be specific, we encode the node importance according to its *degree*, that is we assign each node with a learnable embedding c_i based on its degree, and the weights are randomly initialized, as the degree is a simple measurement of importance in the GED task.

Position encoding. The positional information (p_i) is then encoded into the model, as the nodes located with similar local positions are more likely to match. For instance, the node u_2 is more likely to match the node v_3 as they have similar one-hop and two-hop neighbors in Fig. 1. Shortest-path-distances Ying et al. (2021), PageRank Mialon et al. (2021) and random walk Dwivedi et al. (2022); Li et al. (2020) are generally used to measure the relative position of nodes. From the aspect of low-complexity, we employ the probabilities random walk of different steps as the relative position encoding $p_i \in \mathbb{R}^t$.

$$p_i = [R_{ii}^{(1)}, R_{ii}^{(2)}, \cdots, R_{ii}^{(t)}] \tag{1}$$

where $R = AD^{-1}$ is the the random walk operator, t is the step of number of random walks, and $R_{ii}^{(t)}$ refers to the landing probability of the node i to itself in the t-*th* step of random walk. In such a random walk diffusion manner, we encode the relative positional relationship between a node and its t-hop neighbors from a fine granularity.

Finally, the initial node embeddings $h_i^0 \in \mathbb{R}^d$ are built through a multilayer perceptron (MLP) by concatenating (1) node features x_i, (2) importance encoding c_i, and (3) position encoding p_i.

$$h_i^0 = \texttt{MLP}(x_i \oplus c_i \oplus p_i), \;\; \forall i \in \mathcal{V} \tag{2}$$

Datasets	Structure Operations			Attribution Operations
	Node Insertion	Edge Insertion	Edge Deletion	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%

Table 1: Statistics of types of edit operations. We randomly sample $1,000$ graphs for each datasets and compute their edit operations.

GNN backbone. We adopt the GCN Kipf & Welling (2016) as the backbone of SEGCN 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:

$$h_i^{(l)} = \sigma\Big(\sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} h_j^{(l-1)} w^{(l-1)}\Big) \tag{3}$$

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 due to the different numbers of neighbors, the aggregated embeddings from adjacent nodes are also normalized by the total number of adjacent nodes c_{ij}. SEGCN takes the obtained h_i^0 as the input embedding. After multiple layers of GCNs, SEGCN enhances the learned embeddings by both initial structure information and higher-order neighbor information. Hence, SEGCN combines the property of structure-dominant operations and the heuristic strategy for matching nodes, which is tailored to the GED task.

By Proposition 1, for a given graph pair $\mathcal{G}_1, \mathcal{G}_2$, we have $|\mathcal{V}_1| \leq |\mathcal{V}_2|$. After the encoding by SEGCN, the node embeddings of $\mathcal{G}_1$ and $\mathcal{G}_2$ are denoted as $\mathbf{h}_1 \in \mathbb{R}^{|\mathcal{V}_1| \times d}$ and $\mathbf{h}_2 \in \mathbb{R}^{|\mathcal{V}_2| \times d}$, respectively.

4.2 MATCHING MODULE

Matching module targets to learn the node matchings based on the obtained node embeddings $\mathbf{h}_1$ and $\mathbf{h}_2$ from SEGCN. Due to the inherent equivalence between GED computation and node matching, two complementary learning tasks *i.e.,* learning GED and learning node matchings are put forward, where learning GED mainly focuses on learning the distance between graph representations that assist the node matching task.

Learning GED**.** The GED actually measures the overall similarity of the graph pairs, and we apply graph-level representations to learn GED. In order to highlight the different contribution of each node *w.r.t.*GED, the attention mechanism is designed to derive the graph-level representation.

Specifically, it first generates a graph view $\mathbf{v} \in \mathbb{R}^{1\times d}$, which is the average of node embeddings followed by a nonlinear transformation:

$$\mathbf{v} = \mathtt{tanh}(\bar{\mathbf{h}}_1 \mathbf{W_g}) \tag{4}$$

where $\bar{\cdot}$ is $\mathtt{mean}$ operation for the node embeddings, and the $\mathbf{W_g} \in \mathbb{R}^{d\times d}$ is a learnable attention weight matrix. Then the graph-level representation $\mathbf{h}_1^g \in \mathbb{R}^{1\times d}$ is computed by the weighted sum of the node embeddings where the weight is the attention from the node to the graph view:

$$\mathbf{h}_1^g = \sigma(\mathbf{v}\mathbf{h}_1^\top)\mathbf{h}_1 \tag{5}$$

The graph-level representation $\mathbf{h}_2^g$ of $\mathcal{G}_2$ is also obtained by the attention mechanism with shared weight $\mathbf{W_g}$. The normalized GED (by Equation 11) of $\mathcal{G}_1$ and $\mathcal{G}_2$, *i.e.,* $d_{\mathcal{G}_1,\mathcal{G}_2}$ is predicted using the $\mathtt{MLP}$ operation which gradually reduces the concatenated graph-level representation $\mathbf{h}_1^g$ and $\mathbf{h}_2^g$.

$$d_{\mathcal{G}_1,\mathcal{G}_2} = \mathtt{MLP}(\mathbf{h}_1^g \oplus \mathbf{h}_2^g) \tag{6}$$

Learning node matchings. By Proposition 2, we learn node matchings from fine-grained correspondences, where a pair-wise node similarity matrix is first build. Then the optimal matching is relaxed to find the node matching that maximizes the similarity of node sets from the similarity matrix. Finally, the top-k matching nodes are generated from the similarity matrix, which interprets the multiple optimal node matchings involving in GED.

Similarity matrix. According to the node embeddings $\mathbf{h}_1$ and $\mathbf{h}_2$, one could easily model the node similarity matrix by $\mathbf{h}_1\mathbf{h}_2^\top$. However, it is too hard to learn the distance of the distribution *w.r.t.* the node embeddings. We model it in a more flexible way, and similarity matrix $\mathbf{S} \in |\mathcal{V}_1| \times |\mathcal{V}_2|$ is:

$$\mathbf{S} = \sigma(\mathbf{h}_1^\top \mathbf{W_n} \mathbf{h}_2) \tag{7}$$

where $\mathbf{W_n} \in \mathbb{R}^{d\times d}$ is a learnable weights matrix that models the cost by transforming the embedding $\mathbf{h}_1$ to $\mathbf{h}_2$. All elements of similarity matrix $\mathbf{S}$ are positive after applying the sigmoid function, and $\mathbf{S}_{i,j}$ measures the similarity between $\mathcal{V}_{1i}$ and $\mathcal{V}_{2j}$, which is aware of the attribute and its local structure information. Different from the operations of padding or resizing the similarity matrix Bai et al. (2020), a size of $|\mathcal{V}_1| \times |\mathcal{V}_2|$ is enough to represent all possible matchings by Proposition 1.

Assignment matrix. In order to find the node matching that maximizes the similarity of node sets, we relax it to a linear assignment problem on the similarity matrix, which is typically solved by Hungarian algorithm Munkres (1957). An efficient and simple algorithm Sinkhorn Cuturi (2013); Fey et al. (2020) is typically employed to learn the probabilities of the matchings, which is an approximate and differentiable (as only matrix multiplication and normalization operators are involved) version of Hungarian algorithm. Specifically, Sinkhorn net takes a non-negative similarity matrix and finally coverts it into an assignment matrix $\mathbf{S_a} \in \mathbb{R}^{|\mathcal{V}_1|\times|\mathcal{V}_2|}$, *i.e.,* sum each row/column equals 1, also called a doubly-stochastic matrix.

$$\mathbf{S_a} = \texttt{Sinkhorn}(\mathbf{S}) \tag{8}$$

where it iteratively performs row-normalization, *i.e.,* element-wise division by the sum of its row and column-normalization until convergence. And hence, the element $\mathbf{S_a}_{i,j}$ of the assignment matrix measures the probabilities of $\mathcal{V}_{1i}$ and $\mathcal{V}_{2j}$ belonging to the optimal matching.

Top-k candidate matching nodes. It is natural to find top-k similarity nodes as candidate matching nodes to be aware of the property of multiple optimal node matchings.

During the testing, indeed, one can repeatedly run Hungarian for k times to find optimal k matching nodes for each node. However, it is time-consumable as Hungarian runs in cubic time, *i.e.,* $\mathcal{O}(kn^3)$ time in total. We further propose a greedy Algorithm 1 to find top-k candidate nodes, which runs in $\mathcal{O}(kn^2)$ time. In brief, it iteratively finds a node with the largest matching probability as a candidate node from the *unmatched nodes*, where the injection constraint of node matchings is also guaranteed. Detailed description and analyses of Algorithm 1 present in Appendix B.

Loss design. MATA* is trained in a supervised manner for graph pairs $\mathcal{G}_x$ and $\mathcal{G}_y$ using normalized ground-truth GED $d^t_{x,y}$ by Equation 11 and its corresponding node matching $\mathcal{M}^t_{x,y}$. And the loss function evaluates both the difference for learning GED from the predicted normalized GED $d_{x,y}$ and predicted node matchings from the assignment matrix $\mathbf{S_a}$.

For learning GED task, we minimize the MSE loss:

$$\mathcal{L}_g = \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} (d_{x,y} - d^t_{x,y})^2 \tag{9}$$

where $\mathcal{D}$ is the set of training graph pairs. For learning node matching task, we minimize the negative log-likelihood of the node matchings on the assignment matrix:

$$\mathcal{L}_n = -\frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \sum_{(i,j)\in\mathcal{M}^t_{x,y}} \texttt{log}\mathbf{S_a}_{i,j} \tag{10}$$

Our final loss function is a combination of the above two: $\mathcal{L} = \mathcal{L}_g + \mathcal{L}_n$

Note that, different from the use of permutation cross-entropy loss Wang et al. (2019) or Hungarian loss Yu et al. (2020) 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 lies in that multiple optimal node matchings typically exist, and these unmatched node pairs may also belong to other node matchings corresponding to the GED.

4.3 Optimal Finding Module

MATA* finally integrates A*LSa algorithm Chang et al. (2020) to find the optimal edit distance (*i.e.,* node matching) among the learned top-k candidate matching nodes, where A*LSa conducts a best-first search after treating it as a pathfinding problem to compute GED. The search procedure of MATA* is also pruned by the theoretical bounded estimation of unmatched subgraphs of A*LSa.

For example, a search tree of A*LSa on partial matching nodes is illustrated at the right of Fig. 2, ellipses represent node matching relationship. As can be seen, MATA* only executes on the candidate matching nodes (solid line) to find the optimal solution among these, and the unmatching nodes (dashed line) are pruned from the search tree. That is MATA* performs the learned promising search directions *w.r.t.* node matchings from data to efficiently achieve approximate GED computation.

Note that, MATA* actually finds the sub-optimal solutions of GED computation. The detailed description of A*LSa used in MATA* is demonstrated in Appendix C, and the difference between MATA* and other hybrid approaches are presented in Appendix A

5 Experiments

5.1 Experimental Settings

Datasets. In this work, three benchmark datasets *i.e.,* AIDS Bai et al. (2019), IMDB Yanardag & Vishwanathan (2015), and CANCER [1] are employed. (1) AIDS is a set of antivirus screen chemical compounds labeled with 29 types. Following Bai et al. (2019); Wang et al. (2021), 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 an 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 5.

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 Neuhaus et al. (2006), Hungarian Riesen & Bunke (2009) and VJ Fankhauser et al. (2011). (2) The second category includes two common-used and one state-of-the-art learning models, *i.e.,*, SimGNN Bai et al. (2019), GMN Li et al. (2019) and GENN Wang et al. (2021). (3) We chose an up-to-date model GENNA* as the representative of the third category, and our MATA* also belongs to this category.

Evaluation metrics. We adopt the following experimental metrics to evaluate the performance of the various approaches, and the ground-truth is normalized by Equation 11. (1) Edit path means whether a method can recover the edit path. (2) Accuracy (ACC), (3) Mean Absolute Error (MAE), (4) Mean Squared Error (MSE), (5) Precision at k (p@10), and (6) p@20 are the common used metrics. (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, which records the average running time per graph pair. Refer to Appendix D for details.

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 Bai et al. (2019). 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.

More details of experimental settings and the hyper-parameters of different approaches are presented in Appendix D. The source codes and data are available at `https://anonymous.4open.science/r/mata`.

[1]https://cactus.nci.nih.gov/download/nci/CAN2DA99.sdz

Datasets	Methods	Edit Path	ACC ↑	MAE ↓	MSE ↓	p@10 ↑	p@20 ↑	ρ ↑	τ ↑
AIDS	A*Beam	✓	16.68	0.092	1.37	0.460	0.470	0.720	0.546
	Hungarian	✓	4.19	0.194	4.77	0.293	0.328	0.541	0.386
	VJ	✓	0.95	0.216	5.64	0.215	0.273	0.543	0.387
	SimGNN	×	0.01	0.036	0.22	0.470	0.540	0.886	0.725
	GMN	×	0.02	0.034	0.19	0.401	0.489	0.750	0.673
	GENN	×	0.02	0.031	**0.17**	0.441	0.525	**0.898**	**0.738**
	GENNA*	✓	20.05	0.034	0.46	0.407	0.556	0.515	0.378
	MATA*	✓	**65.20**	**0.027**	0.32	**0.542**	**0.569**	0.856	0.723
IMDB	A*Beam	✓	23.18	0.111	5.22	0.464	0.527	0.489	0.381
	Hungarian	✓	22.53	0.115	5.38	0.438	0.498	0.465	0.359
	VJ	✓	22.24	0.115	5.38	0.436	0.495	0.465	0.359
	SimGNN	×	0.11	0.114	**5.01**	0.474	0.531	0.500	0.388
	GMN	×	0.29	0.128	**5.01**	0.479	0.542	0.513	0.392
	GENN	×	0.22	0.108	5.04	0.476	0.533	0.495	0.384
	GENNA*	✓	–	–	–	–	–	–	–
	MATA*	✓	**44.72**	**0.098**	**5.01**	**0.504**	**0.569**	**0.540**	**0.454**
CANCER	A*Beam	✓	44.23	0.053	**1.14**	0.161	0.266	0.446	0.352
	Hungarian	✓	2.19	0.162	3.56	0.123	0.227	0.139	0.096
	VJ	✓	0.00	0.184	4.85	0.095	0.187	0.188	0.133
	SimGNN	×	0.01	0.068	1.42	0.273	0.297	0.277	0.191
	GMN	×	0.00	0.071	1.47	0.280	0.285	0.254	0.174
	GENN	×	0.00	0.069	1.44	0.285	0.264	0.300	0.207
	GENNA*	✓	–	–	–	–	–	–	–
	MATA*	✓	**55.58**	**0.040**	**1.14**	**0.817**	**0.824**	**0.726**	**0.621**

Table 2: Effectiveness evaluations. The metrics are calculated on the normalized edit distance by Equation 11. Top-k are set to 6, 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.

5.2 EXPERIMENTAL RESULTS

In this section, we evaluate the performance of MATA* from the effectiveness, scalability, efficiency and ablation study. More experiments about top-k comparisons are demonstrated in Appendeix D.

Effectiveness evaluations. Table 2 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 (12.4%, 9.1%, 24.5%). (2) Only learning-based models can not 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 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 (45.2%, 21.5%, 11.4%), 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 of GED 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.

	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

Table 3: Efficiency evaluations. Average running time for solving one graph pair on test data (ms). The training time of learning-based and hybrid approaches do not included.

Scalability *w.r.t.* graph size. Consider the overall performance of the same approach 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 2. (2) Combinatorial 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 $\mathcal{O}(n^2d + d^2n)$ time for each search as explained in the introduction.

Efficiency *w.r.t.* running time. The efficiency of eight approaches on three real-world datasets is reported in Table 3. 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 our 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*.

Datasets	GNN backbones	ACC ↑	MAE ↓	p@10 ↑	ρ ↑
AIDS	SEGCN	**65.20**	**0.027**	**0.542**	**0.856**
	GCN	62.03	0.029	0.523	0.853
IMDB	SEGCN	**44.72**	**0.098**	0.504	**0.540**
	GCN	44.60	**0.098**	**0.510**	0.537
CANCER	SEGCN	**55.58**	**0.040**	0.817	**0.726**
	GCN	52.87	0.043	**0.825**	0.692

Table 4: Ablation study: SEGCN vs. GCN

Ablation study *w.r.t.* structural information encoding. We perform an ablation study on the importance of designs in our proposed SEGCN on AIDS, IMDB, and CANCER. We re-train a model for each dataset which has the same parameter settings and only replaces SEGCN by GCN, and the ablation results are reported in Table 4. From the table, the model with SEGCN outperforms the counterpart built on GCN on ACC, MAE and ρ metrics, *e.g.,* ACC is improved by (3.2%, 0.1%, 2.7%) on (AIDS, IMDB, CANCER), respectively. This demonstrates the effectiveness of using importance encoding and position encoding to capture graph structural information *w.r.t.* structure-dominant editing operations.

6 CONCLUSION

We have presented a data-driven hybrid approach MATA* based on Graph Neural Networks (SEGCN) and A* algorithms, which leverage the learned candidate matching nodes to prune unpromising search directions of A*LSa algorithm to approximate GED. We model it from a new perspective of node matching and combine the intrinsic relationship between GED computation and node matching. Besides, the design of our hybrid approach MATA* is aware of the two combinatorial properties involved in GED computation: structure-dominant operations and multiple optimal node matchings. Finally, we conduct extensive experiments on AIDS, IMDB, and CANCER to demonstrate the effectiveness, scalability, and efficiency of combinatorial search-based, learning-based and hybrid approaches.

References

Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. An exact graph edit distance algorithm for solving pattern recognition problems. In *ICPRAM*, pp. 271–278, 2015.

Jiyang Bai and Peixiang Zhao. Tagsim: Type-aware graph similarity learning and computation. *Proc. VLDB Endow.*, 15(2):335–347, 2021.

Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn: A neural network approach to fast graph similarity computation. In *WSDM*, pp. 384–392, 2019.

Yunsheng Bai, Hao Ding, Ken Gu, Yizhou Sun, and Wei Wang. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. In *AAAI*, pp. 3219–3226, 2020.

David B. Blumenthal and Johann Gamper. Exact computation of graph edit distance for uniform and non-uniform metric edit costs. In *GbRPR*, volume 10310, pp. 211–221, 2017.

David B. Blumenthal and Johann Gamper. On the exact computation of the graph edit distance. *Pattern Recognit. Lett.*, 134:46–57, 2020.

David B. Blumenthal, Nicolas Boria, Johann Gamper, Sébastien Bougleux, and Luc Brun. Comparing heuristics for graph edit distance computation. *VLDB J.*, 29(1):419–458, 2020.

Garcia-Hernandez Carlos, Alberto Fernández, and Francesc Serratosa. Ligand-based virtual screening using graph edit distance as molecular similarity measure. *J. Chem. Inf. Model.*, 59(4):1410–1421, 2019.

Lijun Chang, Xing Feng, Xuemin Lin, Lu Qin, Wenjie Zhang, and Dian Ouyang. Speeding up GED verification for graph similarity search. In *ICDE*, pp. 793–804, 2020.

Lijun Chang, Xing Feng, Kai Yao, Lu Qin, and Wenjie Zhang. Accelerating graph similarity search via efficient ged computation. *IEEE Trans. Knowl. Data Eng.*, pp. Accepted, 2022.

Xiaoyang Chen, Hongwei Huo, Jun Huan, and Jeffrey Scott Vitter. An efficient algorithm for graph edit distance computation. *Knowl. Based Syst.*, 163:762–775, 2019.

Minsu Cho, Karteek Alahari, and Jean Ponce. Learning graphs to match. In *ICCV*, pp. 25–32, 2013.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, pp. 2292–2300, 2013.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. pp. Accepted, 2022.

Stefan Fankhauser, Kaspar Riesen, and Horst Bunke. Speeding up graph edit distance computation through fast bipartite matching. In *GbRPR*, volume 6658, pp. 102–111, 2011.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *ICLR-W*, 2019.

Matthias Fey, Jan Eric Lenssen, Christopher Morris, Jonathan Masci, and Nils M. Kriege. Deep graph matching consensus. In *ICLR*, 2020.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2):100–107, 1968.

Jongik Kim, Dong-Hoon Choi, and Chen Li. Inves: Incremental partitioning-based verification for graph similarity search. In *EDBT*, pp. 229–240, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

Danai Koutra, Hanghang Tong, and David Lubensky. Big-align: Fast bipartite graph alignment. In *ICDM*, pp. 389–398, 2013.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. In *NeurIPS*, 2020.

Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *ICML*, volume 97, pp. 3835–3845, 2019.

Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers. *CoRR*, abs/2106.05667, 2021.

James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

Michel Neuhaus, Kaspar Riesen, and Horst Bunke. Fast suboptimal algorithms for the computation of graph edit distance. In *IAPR Workshops*, volume 4109, pp. 163–172, 2006.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Yun Peng, Byron Choi, and Jianliang Xu. Graph edit distance learning via modeling optimum matchings with constraints. In *IJCAI*, pp. 1534–1540, 2021.

Kaspar Riesen and Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image Vis. Comput.*, 27(7):950–959, 2009.

Kaspar Riesen, Stefan Fankhauser, and Horst Bunke. Speeding up graph edit distance computation with a bipartite heuristic. In *MLG*, 2007.

Kaspar Riesen, Sandro Emmenegger, and Horst Bunke. A novel software toolkit for graph edit distance computation. In *GbRPR*, volume 7877, pp. 142–151, 2013.

Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *ICCV*, pp. 3056–3065, 2019.

Runzhong Wang, Tianqi Zhang, Tianshu Yu, Junchi Yan, and Xiaokang Yang. Combinatorial learning of graph edit distance via dynamic embedding. In *CVPR*, pp. 5241–5250, 2021.

Junchi Yan, Shuang Yang, and Edwin R. Hancock. Learning for graph matching and related combinatorial optimization problems. In *IJCAI*, pp. 4988–4996, 2020.

Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. In *SIGKDD*, pp. 1365–1374. ACM, 2015.

Lei Yang and Lei Zou. Noah: Neural-optimized a* search algorithm for graph edit distance computation. In *ICDE*, pp. 576–587, 2021.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *NeurIPS*, pp. 28877–28888, 2021.

Tianshu Yu, Runzhong Wang, Junchi Yan, and Baoxin Li. Learning deep graph matching with channel-independent embedding and hungarian attention. In *ICLR*, 2020.

A Appendix: Detailed Related Works.

In this section, we first present a detailed overview of existing literature from three categories: combinatorial search-based, learning-based and hybrid GED computation. We then classify the connections and differences between MATA* and related studies.

Combinatorial search-based. Combinatorial search-based algorithms either directly explore the search space corresponding to GED, or relax GED computation to other combinatorial problems with polynomial time complexity, *e.g.,* bipartite matching problems. Combinatorial search-based are developed from the exact and approximate.

Exact computation. A* search-based algorithms are widely used for exact GED computation Riesen et al. (2007); Hart et al. (1968), which treat it as a pathfinding problem and perform best-search to expand the search path Yang & Zou (2021). Specifically, all possible matchings (*w.r.t.* the edit operations) between two graphs are represented by a search tree, and search direction is explored by the best-search, *i.e.,* determined by the cost of matched subgraphs and estimation cost of unmatched subgraphs. Different A*-based algorithms mainly focus on how to better estimate the cost of unmatched subgraphs with the theoretical guarantee, such as using label sets Riesen et al. (2007; 2013) or subgraph structures Chang et al. (2020; 2022); Kim et al. (2019). Besides, there are some efforts of exploring the search tree in a depth-first search fashion to improve efficiency Abu-Aisheh et al. (2015); Blumenthal & Gamper (2017).

Approximate GED *computation.* Due to the search space of exact solutions growing factorially with the number of nodes, different approximations are proposed to find the sub-optimal solutions. Specifically, A*Beam is proposed based on A* search, which only explores the most possible directions with limited beam size Neuhaus et al. (2006). Besides, GED computation is relaxed to bipartite matching to trade off the efficiency and accuracy, where only the local structure is considered, and hence the Hungarian Riesen & Bunke (2009) and VJ Fankhauser et al. (2011) are utilized to find the optimal bipartite matching in $\mathcal{O}(n^3)$ time.

Learning-based GED **computation.** With the progress of graph representation techniques of Graph Neural Networks Kipf & Welling (2016); Ying et al. (2021); Dwivedi et al. (2022), 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 GED computation task. Bai et al. (2019) first presents a model using GCN Kipf & Welling (2016) and attention mechanism to approximately learn GED in an end-to-end fashion. Based on Bai et al. (2019), Bai et al. (2020) further introduces a multi-scale node comparison technique to extract the fine-grained information from the node-to-node similarity matrix. Besides, Li et al. (2019) incorporates both the node and graph level information by the cross-graph module to trade-off the accuracy & computation. Bai & Zhao (2021) splits the graph edit distance into different types of edit operations and applies graph aggregation layers to learn each type individually. More recently, Peng et al. (2021) designs a GED-specific regularizer to impose the matching constraints involved in GED, where the graph pairs are represented by the association graphs.

Hybrid GED **computation.** Recently, by combining the progress of deep learning and combinatorial-search techniques, some works that combine GNNs and A* algorithms have been proposed Wang et al. (2021); Yang & Zou (2021). Actually, these two papers employee GNNs to optimize the search directions of A* algorithms, both of which predict the estimation cost of unmatched subgraphs during A* search algorithm. Specifically, Yang & Zou (2021) proposes graph path networks incorporating pre-training edit path information and cross-graph information for training the model, and optimizes A* by predicting the cost of unmatched subgraphs using graph path networks and beam size to prune the search of A* algorithm. Wang et al. (2021) integrates a dynamic graph embedding network for predicting the estimation cost of unmatched subgraphs, where the network is improved from Bai et al. (2019).

Connections & differences. We highlight the connections & differences between our hybrid MATA* and learning-based, existing hybrid approaches from the (1) formulation and effectiveness, (2) efficiency and scalability.

(1) Existing hybrid methods modeling the cost of unmatched subgraphs (*i.e.,* edit distances) in A* algorithms as a regression problem. That is learning-based and hybrid approaches all explicitly and

Algorithm 1 Greedy matching

Input: The assignment matrix $\mathbf{S_a}$, and k
Output: Top-k matching nodes matk

```
1:  $t = 0$
2:  while $t < k$ and $t = t + 1$ do
3:      $\mathsf{cols} = \emptyset$  ▷ store all column ids of a node matching.
4:      for all $i$ in $\mathcal{V}_1$ do
5:          Find the greatest $\mathbf{S_a}_{i,j}$ with $j$ not in cols;
6:          Insert $\mathbf{S_a}_{i,j}$ into $\mathsf{matk}[t]$, and set $\mathbf{S_a}_{i,j}$ to 0;
7:          Update cols according to $j$;
8:      end for
9:  end while
10: return matk.
```

implicitly model GED as a regression problem, respectively. Hence, the intrinsic equivalence between the GED computation and node matching is not captured by their approximate GED models, which limits the effectiveness of the algorithms.

(2) GNNs with the attention mechanism are employed to estimate the cost function for Wang et al. (2021) and Yang & Zou (2021), which take $\mathcal{O}(n^2d + d^2n)$ time for extending each search, and encounter scalability issues. Besides, due to the inaccurate GED approximation in the cost function estimation of A*, the search space of Wang et al. (2021) and Yang & Zou (2021) all factorially increase with the number of nodes in the worst case. And hence, Wang et al. (2021) even runs slower than the exact GED computation Chang et al. (2022). These methods are also difficult to scale to large datasets. Due to the polynomial time complexity of the learning-based algorithms, they run efficiently for approximate GED computation via an end-to-end manner. However, the scalability heavily relies on the ground-truth produced by combinatorial search-based algorithms.

B APPENDIX: GREEDY TOP-k CANDIDATE NODES

Finding top-k candidate nodes from the assignment matrix is presented in Algorithm 1.

Specifically, it takes the assignment matrix $\mathbf{S_a}$ and k as input, and returns the top-k matching nodes matk. It first initializes t to 0 and iteratively executes t for k times (lines 1, 2-9). In each iteration, it initializes cols to an empty set, which stores the selected element's column id of a node matching (*i.e.,* the corresponding node of $\mathcal{G}_2$). cols is also used to guarantee the injection constraint (line 3). Then, for each row of $\mathbf{S_a}$ (*i.e.,* each node of $\mathcal{G}_1$), it finds the largest element $\mathbf{S_a}_{i,j}$ with its column id j not in cols. $\mathbf{S_a}_{i,j}$ is inserted into $\mathsf{matk}[t]$, and cols is updated according to j. That is node j with the largest matching probability with node i is selected for the candidate set. In such a way, a candidate node mapping is found (lines 4-8). Once k iterations are completed, the top-k matching nodes matk are returned (line 10).

It is easy to know that Algorithm 1 takes $\mathcal{O}(kn^2)$ time, where n is the maximum number of nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$.

C APPENDIX: DETAILED REVIEW OF A* FOR GED

We first present the idea of A* for computing GED, then review the recent algorithm A*LSa, which is the most efficient both due to its powerful pruning strategy and low computation cost of the strategy. Finally, we demonstrate how to integrate A*LSa into our hybrid MATA* to find the optimal solution among top-k candidate matching nodes.

A*-based GED **computation.** A* algorithms treat it as a pathfinding problem and perform the best-search to expand the search path. Specifically, given two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, all possible node matchings, *i.e.,* the search space of GED can be organized as a search tree $\mathcal{T}$. The root of $\mathcal{T}$ represents to start constructing $\mathcal{T}$. The level i of the $\mathcal{T}$ refers to the possible matchings between the $i\text{-}th$ node of ord and all unmatched nodes of $\mathcal{G}_2$, and each node of $\mathcal{T}$ refers to a matching node pair from $\mathcal{G}_1$ to $\mathcal{G}_2$ (*e.g.,* $\mathcal{V}_{1i}$ to $\mathcal{V}_{2j}$). The parent-child relationships is naturally defined based on the

Algorithm 2 A*LSa Chang et al. (2020)

```
Input: Graphs $\mathcal{G}_1$ and $\mathcal{G}_2$
Output: ged($\mathcal{G}_1, \mathcal{G}_2$)
 1: Compute the matching order ord of $\mathcal{V}_1$;
 2: Push $(0, \emptyset, null, 0)$ into $Q$;        ▷ Initialize the priority queue $Q$ by the root of $\mathcal{T}$.
 3: while $Q \neq \emptyset$ do
 4:     Pop $(i, f, pa, lb)$ with minimum $lb$ from $Q$;
 5:     Compute the lower bound $lb$ for each child $c$ of $f$;
 6:     for all child $c$ of $f$ do
 7:         if $i + 1 = |\mathcal{V}_1|$ then ged($\mathcal{G}_1, \mathcal{G}_2$) = $c.lb$; break;
 8:         else Push $(i + 1, c, f, lb)$ into $Q$;
 9:         end if
10:     end for
11: end while
12: return ged($\mathcal{G}_1, \mathcal{G}_2$)
```

search tree $\mathcal{T}$. And hence, a path from the root to leaf nodes is a node matching, *i.e.,* all nodes of $\mathcal{G}_1$ are mapped to nodes of $\mathcal{G}_2$. A* algorithms indeed perform on the search tree to find a path from the root to leaf nodes, where the directions are determined by the *lower bound*. The lower bound of a partial node matching in $\mathcal{T}$ is defined as the sum of the edit cost of matched subgraphs and the estimated cost of unmatched subgraphs. Note that, the GED can be easily computed by scanning the $\mathcal{G}_1$ and $\mathcal{G}_2$ one time if a node matching is obtained Chang et al. (2020).

Review of A*LSa. A*LSa Chang et al. (2020) conducts a best-first search on the search tree $\mathcal{T}$ by computing the theoretical bounded estimation of unmatched subgraphs to exact compute GED, shown in Algorithm 2. A priority queue Q is maintained to store the search states (nodes of $\mathcal{T}$) during the process, which contains its level i, current partial matching f, its parent matching pa and its lower bound lb. After the matching order ord is computed, A*LSa initializes the priority queue Q by the root of $\mathcal{T}$ (lines 1, 2). It then iteratively pops (i, f, pa, lb) from Q with the minimum lower bound, and extends the current matching f by computing the lower bound of each child (lines 3-6, 8). If the full node matching is formed, then ged($\mathcal{G}_1, \mathcal{G}_2$) equals its lower bound and is returned (lines 7, 12).

For estimating the cost of the unmatched subgraphs, A*LSa proposes an anchor-aware label set-based method to make use of the information of the matched nodes and their anchor sets, which computes the lower bound cost for all children of f in $\mathcal{O}(\mathcal{E}_1 + \mathcal{E}_2)$ time Chang et al. (2020).

Integrate A*LSa to MATA*. For testing in MATA*, top-k candidate matchings are generated from the assignment matrix using Algorithm 1. These matchings are first translated to the node of the search tree $\mathcal{T}$ in the initialization of A*LSa. During the search procedure, for each child node extending from the parent node in the search tree $\mathcal{T}$, MATA* only adds the child node corresponding to the matchings into the priority queue Q, and the other child nodes are directly pruned. Note that, we do not modify the unmatched subgraph cost estimation of A*LSa.

D APPENDIX: DETAILED EXPERIMENTAL SETTINGS & RESULTS

In this section, we present the detailed experimental settings and the performance *w.r.t.* different top-k selection.

D.1 DETAILED EXPERIMENTAL SETTINGS.

Dataset processing. For each dataset, we generate the graph edit distance (or sub-optimal edit distance) and its corresponding node matching of the graph pairs. Specifically, for AIDS, all graphs are less than ten nodes, where the GED and exact node matching of all graph pairs are computed by the exact algorithm A*LSa Chang et al. (2020). For the larger datasets IMDB and CANCER, the exact solutions are unavailable as they are not reliable to compute in a reasonable time for more than 16 nodes Blumenthal & Gamper (2020). Following Bai et al. (2019) and Peng et al. (2021), we choose the smallest edit distance and its corresponding node matching computed by the three

	#Graphs	#Pairs	avg($|\mathcal{E}|/|\mathcal{V}|$)	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

Table 5: Statistics of three real-life datasets. The graph pairs are partitioned 60%, 20%, 20% as training, validation and test sets, respectively.

approximate algorithms (*i.e.,*, A*Beam, Hungarian, VJ) as ground-truth. The ground-truth distance is also denoted by $\mathsf{ged}(\cdot,\cdot)$ for simplicity. Further, the computed distance is normalized by

$$\exp\{-\mathsf{ged}(\mathcal{G}_1,\mathcal{G}_2)\times 2/(\mathcal{V}_1+\mathcal{V}_2)\}, \tag{11}$$

which transforms the $\mathsf{ged}(\cdot,\cdot)$ into a similarity score in the range of $(0,1]$.

In such a manner, we compute the ground-truth of graph pairs in (AIDS, IMDB, CANCER) with a number of (490K, 2.25M, 100K), respectively, and we randomly partition 60%, 20%, and 20% of graph pairs as training, validation and test sets for each dataset.

Parameter settings. We conduct all experiments on machines with Intel Xeon Gold@2.40GHz CPU and NVIDIA Tesla V100 32GB GPU. Our model MATA* is implemented in PYTORCH Paszke et al. (2017) using the PYTORCH GEOMETRIC Fey & Lenssen (2019) and A*LSa algorithm is implemented in C for the performance consideration, and is open-source. The number of SEGCN layers is set to 3 and ReLU is utilized as the activation function, where the number of feature channels of the three layers is all set to 64. The dimensions of learnable embedding c_i for importance encoding are set to 12, 40 and 18 for AIDS, IMDB and CANCER, respectively. The random walk step t is set to 16 for the three datasets. The beam size of A*LSa is set to in a range of $[2, 1000]$ for IMDB and CANCER by rule. In the training phase, we set the batch size to 128 and use Adam optimizer Kingma & Ba (2015) with 0.001 learning rate and 5×10^4 weight decay for each dataset. In the testing phase, we set top-k to 6, 6 and 8 for AIDS, IMDB and CANCER, respectively.

We use the open-source implementation of three combinatorial search-based algorithms (A*Beam, Hungarian and VJ), which is a Java software toolkit Riesen et al. (2013). Along the setting with Bai et al. (2019) and Peng et al. (2021), we use the default configuration of the toolkit, where the beam size is set to 10 for A*Beam. We also modify the implementation of SimGNN, GMN, GENN and GENNA* to support CANCER dataset with their recommendation configurations. The batch sizes of SimGNN and GENN are set to 128, and the batch size GENNA* and GMN are set to 1. Note that, the experiments of GENNA* in their paper were only conducted on the datasets with no more than 10 nodes graphs Wang et al. (2021), and the official implementation of GENNA* also fails to run the larger datasets IMDB and CANCER due to memory overflow on 32GB machine or running with more than 10 minutes for one graph pair.

Metrics (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. And the algorithms involving learning only report the testing time.

D.2 PERFORMANCE *w.r.t.* TOP-k SELECTION.

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

Datasets	top-k	ACC ↑	MAE ↓	MSE ↓	p@10 ↑	p@20 ↑	ρ ↑	τ ↑	Time ↓
AIDS	5	54.22	0.042	0.584	0.469	0.497	0.767	0.624	**4.3**
	6	65.20	0.027	0.320	0.542	0.569	0.856	0.723	4.4
	7	75.93	0.016	0.166	0.614	0.649	0.918	0.807	4.4
	8	84.17	0.009	0.083	0.695	0.748	0.956	0.868	4.5
	9	90.08	0.005	0.038	0.787	0.825	0.977	0.910	4.8
	10	**100.00**	**0.000**	**0.000**	**1.000**	**1.000**	**1.000**	**1.000**	5.2
IMDB	5	38.57	0.106	5.65	0.381	0.511	0.578	0.472	**30.2**
	6	39.40	0.105	5.61	0.391	0.524	0.579	0.475	35.3
	7	40.97	0.103	5.55	0.401	0.529	0.582	0.480	43.3
	8	41.56	0.102	5.52	0.400	0.532	0.582	0.482	51.9
	9	44.47	**0.100**	5.47	0.412	0.536	0.587	0.491	53.2
	10	**45.05**	**0.100**	**5.45**	**0.415**	**0.541**	**0.588**	**0.492**	55.3
CANCER	5	5.01	0.104	2.08	0.452	0.431	0.678	0.497	**129.8**
	6	7.37	0.091	1.80	0.486	0.497	0.709	0.528	146.8
	7	10.55	0.079	1.56	0.543	0.553	0.734	0.555	153.1
	8	14.37	0.069	1.40	0.569	0.547	0.758	0.582	168.0
	9	22.63	0.058	1.24	0.615	0.627	0.777	0.609	176.7
	10	**34.19**	**0.048**	**1.11**	**0.684**	**0.664**	**0.794**	**0.637**	193.2

Table 6: Performance evaluations *w.r.t.* different top-k. The metrics are calculated in the same way as those in Table 2.

(1) Different 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 small with the exact solution dataset, and MATA* also achieves the optimal solutions running in 5.2 microseconds, 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 2. Actually, k is set to 10 in this test, which achieves the smaller edit distance, and the ground-truth of IMDB and CANCER are further updated by these. Hence, the evaluation metrics are calculated by the updated ground-truth, which produces worse metrics on IMDB and CANCER.