Fused Gromov-Wasserstein Alignment for Graph Edit Distance Computation and Beyond

Jianheng Tang
Hong Kong University of Science and Technology
jtangbf@connect.ust.hk

Xi Zhao
Hong Kong University of Science and Technology
xzhaoca@cse.ust.hk

Lemin Kong
Chinese University of Hong Kong
lkong@se.cuhk.edu.hk

Xiaofang Zhou
Hong Kong University of Science and Technology
zxf@cse.ust.hk

Jia Li*
Hong Kong University of Science and Technology (Guangzhou)
jialee@hkust-gz.edu.cn

ABSTRACT

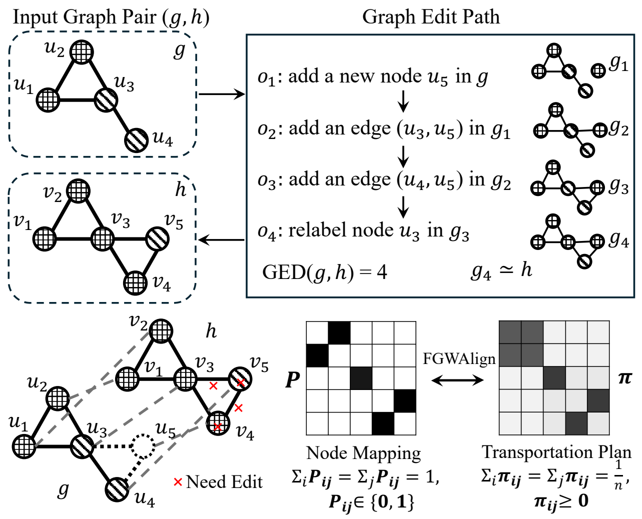
Graph Edit Distance (GED) is a widely recognized metric for measuring graph similarity, yet its NP-complete nature poses challenges for fast and accurate computation. This paper introduces FGWAlign, an Optimal Transport (OT)-based approach for graph alignment and GED computation. We take the first step to theoretically demonstrate and that computing GED can be transformed into optimizing a particular OT variant—the Fused Gromov-Wasserstein distance. Tailored to the GED problem structure, we further implement three key enhancements to the standard FGW solver: (1) a random exploration scheme to better locate the global optimum, (2) a diverse projection strategy for post-processing the transportation plan to escape local optima, and (3) a novel extension to accommodate multi-relational graphs with edge labels. With $O(|V||E|)$ time complexity and $O(|V|^2)$ space complexity, where $|V|$ and $|E|$ are the maximum number of nodes and edges between the two compared graphs, FGWAlign achieves a superior balance of efficiency, accuracy, and scalability. Empirical results show that, compared with 12 representative GED computation methods across different categories on 4 real-world graph datasets, FGWAlign reduces computation errors by over 80% and achieves 15-60× speedup. It also demonstrates promising resutls on downstream applications including labeled graph alignment and graph-level anomaly detection, highlighting its versatility. FGWAlign opens up promising avenues for future applications in graph data management.

PVLDB Reference Format:
Jianheng Tang, Xi Zhao, Lemin Kong, Xiaofang Zhou, and Jia Li. Fused Gromov-Wasserstein Alignment for Graph Edit Distance Computation and Beyond. PVLDB, 18(10): 3641 - 3654, 2025.
doi:10.14778/3748191.3748221

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/squareRoot3/FGWAlign.

* Corresponding Author: Jia Li.

Proceedings of the VLDB Endowment, Vol. 18, No. 10 ISSN 2150-8097.
doi:10.14778/3748191.3748221


Input Graph Pair (g, h)
Graph Edit Path
o_1: add a new node u_5 in g
o_2: add an edge (u_3, u_5) in g_1
o_3: add an edge (u_4, u_5) in g_2
o_4: relabel node u_3 in g_3
GED$(g, h) = 4$
$g_4 \simeq h$
× Need Edit
$\boldsymbol{P}$
FGWAlign
$\boldsymbol{\pi}$
Node Mapping
$\Sigma_i \boldsymbol{P}_{ij} = \Sigma_j \boldsymbol{P}_{ij} = 1$, $\boldsymbol{P}_{ij} \in \{0, 1\}$
Transportation Plan
$\Sigma_i \boldsymbol{\pi}_{ij} = \Sigma_j \boldsymbol{\pi}_{ij} = \frac{1}{n}$, $\boldsymbol{\pi}_{ij} \geq 0$
Figure 1: Example of an edit path for graph pair (g, h) with the corresponding node mappings in bipartite graph matching.

1 INTRODUCTION

As of 2024, the PubChem database holds 119 million chemical compounds and 327 million substances [30], while the AlphaFold database offers over 200 million protein structure predictions [68]. This surge in structured data has intensified challenges in graph data management, particularly in similarity search—identifying graphs resembling a query graph. This process is vital in bioinformatics for protein retrieval [22, 27, 67], multimedia processing for pattern recognition [45, 72], and software engineering for dependency analysis [73], to name a few.

Unlike metrics in vector spaces, quantifying graph similarity presents unique challenges. Traditional approaches include maximum common subgraph detection [9, 79], spectral analysis [74], graph kernel methods [51, 69], and learning-based embeddings [56]. Among these approaches, the *Graph Edit Distance* (GED) stands out as a particularly versatile metric. Analogous to string edit distance [80], GED quantifies the minimum cost required to transform one graph into another through an edit path—a sequence of elementary edit operations. These operations comprise adding or deleting an edge, adding or deleting an isolated node, and relabeling a node or edge. Each operation is assigned a non-negative cost, and the total cost of an edit path is the sum of its constituent operation costs. Figure 1 illustrates an edit path between graphs g and h, comprising four operations (o_1, o_2, o_3, o_4) that transform g into g_4, which is isomorphic to h. Given uniform unit costs for all operations, this minimal-length path yields $\text{GED}(g, h) = 4$.

GED stands out among graph similarity measures due to several key advantages: it fully utilizes graph attributes including node labels and edge types, preserves important metric properties such as triangle inequality, and enables controllable error tolerance in similarity search compared to exact matching. However, these benefits come with substantial computational challenges. Computing exact GED is NP-hard for uniform edit costs [82] and APX-hard for metric edit costs [36]. Moreover, even approximating GED within any ratio is GI-hard, as determining whether GED equals zero is equivalent to solving graph isomorphism [5].

These computational challenges have spawned diverse approaches for GED computation [5, 21, 61]. The predominant strategy reformulates GED computation as a graph alignment problem [58, 60]. As illustrated in the bottom left of Figure 1, a node mapping between graphs g and h naturally defines an edit path by identifying differences between corresponding nodes and edges. While exhaustive search guarantees optimality, its exponential complexity becomes intractable for larger graphs. Heuristic methods incorporating pruning strategies [1, 49] accelerate the search by prioritizing promising partial mappings, but require restrictive beam sizes for practical runtime, often at the cost of solution quality. Alternative formulations based on quadratic assignment [8, 33] or simplified linear assignment [8, 18, 31] often sacrifice global graph structure information, leading to suboptimal results [5].

Recent deep learning approaches have emerged for GED computation, either directly predicting GED values [3, 4, 56] or guiding classical search algorithms [39, 72, 78]. While these methods offer improved accuracy and efficiency, they require substantial ground-truth GED data for training. Such training labels must either come from classical GED computation methods—inheriting their limitations—or from synthetic graph pairs. The latter often leads to poor generalization to real-world scenarios, as demonstrated by TagSim [3], which shows an 11-fold increase in error when moving from synthetic to real graphs. These limitations underscore the need for a more general approach to GED computation.

In this paper, we introduce **FGWAlign**, a novel framework that models GED computation using the Optimal Transport (OT) theory. The key insight of our approach, illustrated in Figure 1, is the relaxation of one-to-one node mappings into probabilistic transportation plans, constrained only by non-negativity. This reformulation offers two fundamental advantages: (1) It transforms the discrete optimization problem into a continuous one, enabling the use of more efficient optimization algorithms, and (2) It allows for a more comprehensive exploration of the solution space, which is particularly valuable given the highly non-convex nature of the GED objective. While OT has shown remarkable success in various structure comparison tasks, including entity matching [16, 52, 65], network alignment [35, 64, 76], and shape correspondence [47, 54, 63], its application to GED computation remains unexplored.

We show that the Fused Gromov-Wasserstein (FGW) distance [66] is well-suited for GED computation due to its ability to jointly compare node labels/attributes and graph structures, by establishing a theoretical connection between the two. We further introduce three key enhancements to boost the standard FGW solver: (1) a diverse projection strategy that efficiently converts the probabilistic transport plan back into multiple candidate discrete node alignments, increasing diversity and helping escape local optima, (2) a randomized initialization scheme that improves the likelihood of finding the global optimum, and (3) an extended formulation that incorporates edge label modifications, enabling GED computation for multi-relational graphs.

FGWAlign achieves superior performance, efficiency, and scalability compared to existing GED computation methods. On 4 real-world datasets, it achieves an over 80% reduction in computation errors and demonstrates superior performance in similarity ranking and edit path generation. Regarding efficiency, FGWAlign attains a 15 to 60 times speedup while maintaining comparable performance with state-of-the-art approaches. With $O(|V|^2)$ space and $O(|V||E|)$ time complexity, FGWAlign computes GED between graphs exceeding 10,000 nodes in under 10 minutes on a single consumer GPU—40 times larger than previous records.

Beyond GED computation, FGWAlign also demonstrates promising results on diverse downstream applications. In labeled or heterogeneous graph alignment tasks across three datasets with 1,000-10,000 nodes, FGWAlign surpasses state-of-the-art methods in structural metrics, particularly in edge correctness. For graph-level anomaly detection, FGWAlign outperforms both classic methods and specialized GNN-based approaches on four of six standard TU datasets. These results highlight FGWAlign's versatility as a unified framework for graph comparison tasks, offering an interpretable yet high-performance alternative to task-specific learning-based methods without requiring significant modifications to the core algorithm.

The remainder of this paper is organized as follows: Section 2 presents the preliminaries and prior work on GED computation and the FGW distance. Section 3 details our proposed FGWAlign method and its three key improvements. Section 4 establishes the theoretical connections between GED and FGW through rigorous proofs. Section 5 reports our experimental findings. Section 6 concludes with a summary and directions for future work.

2 PRELIMINARIES AND RELATED WORK

2.1 Definition of GED

In line with the framework established by Zeng et al. [82], we consider undirected simple graphs without self-loops and multi-edges. A such graph is denoted as $g = (\boldsymbol{V}, \boldsymbol{E}, l)$, where $\boldsymbol{V}$ represents a finite set of nodes, $\boldsymbol{E} \subseteq \boldsymbol{V} \times \boldsymbol{V}$ is a set of node pairs, and $l : \boldsymbol{V} \rightarrow \Sigma^1$ is a labeling function for nodes, with Σ^1 being a finite set of node labels. The adjacency matrix of g is represented as $\boldsymbol{A}$, with $\boldsymbol{A}_{ij} = 1$ indicating the presence of an edge $(v_i, v_j) \in \boldsymbol{E}$, and 0 otherwise.

Graph modification operations, denoted as o, consist of (1) adding or removing an edge, (2) adding or removing an isolated node, and (3)changing the label of a node. An edit path $\mathcal{P}$ is a sequence of edit operations, $\mathcal{P} := (o_i)_{i=1}^{r}$, that transforms a graph g into a graph isomorphic to h:

$$\mathcal{P}(g) = (o_r \circ \cdots \circ o_1)(g) \simeq h. \quad (1)$$

Symbols	Description
$\boldsymbol{V}^g, \boldsymbol{E}^g, \boldsymbol{l}^g$	Node set, edge set, and labeling function of graph g
$\boldsymbol{A}^g$	Adjacency matrix of graph g
$\lvert\boldsymbol{V}^g\rvert, \lvert\boldsymbol{E}^g\rvert$	Number of nodes and edges in graph g
$\mathrm{GED}(g,h)$	Graph Edit Distance between graphs g and h
$f_{\mathrm{GED}}(\cdot, \boldsymbol{P})$	Objective function of GED induced by node mapping
$\boldsymbol{P}$	Permutation matrix representing a node mapping
$f_{\mathrm{FGW}}(\cdot, \boldsymbol{\pi})$	Objective function of the FGW distance
$\boldsymbol{\pi}$	Probabilistic matrix representing a transportation plan
$\langle\cdot,\cdot\rangle$	Frobenius dot product, $\langle A, B\rangle = \sum_{ij} A_{ij}B_{ij}$

Table 1: Frequently used notations in this paper.

Category	Method	Time Complexity	Space Complexity	Guarantee
Search-based	A*-GED [49]	$O(\lvert V\rvert\lvert E\rvert \cdot \lvert V\rvert!)$	$O(\lvert V\rvert \cdot \lvert V\rvert!)$	Optimal
	DF-GED [1]	$O(\lvert V\rvert\lvert E\rvert \cdot \lvert V\rvert!)$	$O(\lvert V\rvert + \lvert E\rvert)$	Optimal
	A*-LSa [11]	$O((\lvert V\rvert + \lvert E\rvert)\lvert V\rvert!)$	$O(\lvert V\rvert \cdot \lvert V\rvert!)$	Optimal
Assignment-based	Star-LB [82]	$O(\lvert V\rvert^3)$	$O(\lvert V\rvert^2)$	Approximate
	Star-UB [82]	$O(\lvert V\rvert^6)$	$O(\lvert V\rvert^2)$	Approximate
	IPFP [33]	$O((\lvert V\rvert + \lvert E\rvert)^2)$	$O((\lvert V\rvert + \lvert E\rvert)^2)$	Heuristic
	RRWM [14]	$O((\lvert V\rvert + \lvert E\rvert)^2)$	$O((\lvert V\rvert + \lvert E\rvert)^2)$	Heuristic
	Hungarian [59]	$O(\lvert V\rvert^3)$	$O(\lvert V\rvert^2)$	Heuristic
Learning-based	SimGNN [4]	$O(\lvert V\rvert^2)$	$O(\lvert V\rvert^2)$	Heuristic
	TaGSim [3]	$O(\lvert V\rvert + \lvert E\rvert)$	$O(\lvert V\rvert + \lvert E\rvert)$	Heuristic
	Noah-A* [78]	$O(\lvert V\rvert\lvert E\rvert \cdot \lvert V\rvert!)$	$O(\lvert V\rvert \cdot \lvert V\rvert!)$	Optimal
	GEDGNN [55]	$O(\lvert V\rvert^3)$	$O(\lvert V\rvert^2)$	Heuristic
	GEDIOT [13]	$O(\lvert V\rvert^3)$	$O(\lvert V\rvert^2)$	Heuristic
Ours	FGWAlign	$O(\lvert V\rvert\lvert E\rvert)$	$O(\lvert V\rvert^2)$	Approximate

Table 2: An overview of GED computation methods categorized by approach, showing worst-case time/space complexity and quality guarantees.

The cost of an edit path, denoted by $c(\mathcal{P}) := \sum_{i=1}^{r} c(o_i)$, is the cumulative cost of its constituent edit operations. This leads to the formal definition of graph edit distance.

Definition 2.1 (GED defined by edit path). *The graph edit distance for graph pair (g, h) is the minimum cost among all feasible edit paths transforming graph g into an isomorphic graph of h:*

$$\mathrm{GED}(g,h) := \min\{c(\mathcal{P}) \mid \mathcal{P}(g) \simeq h\} \tag{2}$$

Converting GED to Graph Alignment. Since enumerating all edit paths is infeasible, GED computation is commonly reformulated as a graph alignment problem [55, 82]. Given graphs g and h where $|\boldsymbol{V}^g| \le |\boldsymbol{V}^h| = n$, we first augment g with $|\boldsymbol{V}^h| - |\boldsymbol{V}^g|$ isolated dummy nodes to equalize the node counts. The node mapping between graphs is then established through a permutation matrix $\boldsymbol{P} \in \{0,1\}^{n\times n}$, where $\boldsymbol{P}_{ij} = 1$ indicates that nodes $u_i \in \boldsymbol{V}^g$ and $v_j \in \boldsymbol{V}^h$ are matched.

Definition 2.2 (GED induced by Graph Alignment). *The GED between graphs g and h can be computed as:*

$$\mathrm{GED}(g,h) = \min_{P\in\mathbb{P}_n} \underbrace{\langle C, \boldsymbol{P}\rangle}_{c_l} + \underbrace{\frac{1}{2}\|\boldsymbol{A}^g - \boldsymbol{P}\boldsymbol{A}^h\boldsymbol{P}^T\|_{1,1}}_{c_e} \tag{3}$$

where $\langle C, \boldsymbol{P}\rangle := \sum_{i,j} C_{ij}\boldsymbol{P}_{ij}$ represents node relabeling costs (C_{ij} being the cost of relabeling node u_i to v_j), $\mathbb{P}_n$ is the set of permutation matrices, and $\|\cdot\|_{1,1}$ denotes the entry-wise l_1 norm.

This formulation transforms GED into an assignment problem with $n!$ feasible solutions. In the next section, we review existing methods for GED computation and discuss their limitations.

2.2 Computation of GED

Table 2 provides a comprehensive overview of existing GED computation methods based on their computational characteristics and solution quality guarantees. It's important to note that: (1) We consider worst-case time and space complexities for each method to establish upper bounds on resource requirements; (2) Many algorithms have different variants with varying complexity profiles, and readers should refer to the original papers for detailed analyses; (3) We focus exclusively on the maximum node size $|\boldsymbol{V}|$ and edge size $|\boldsymbol{E}|$ of the graphs being compared, treating all other parameters as constants for simplified comparison.

2.2.1 *Search-based methods.*

They involve exploring a search tree that represents all possible node mappings between two graphs, organized in a prefix-shared format according to a predefined matching order. To avoid exhaustive search, best-first or A* search methods [11, 49, 59, 85, 86, 88] are used for search space exploration and unpromising subspace pruning. These methods prioritize expanding partial mappings with the lowest estimated cost. Mappings exceeding the beam size are pruned. Typical lower bound cost estimations include the Hungarian/Kuhn-Munkres algorithm [2, 59], the label set heuristic [17, 49], and the anchor-aware estimation [11].

Due to the significant memory demands of A* search, which requires maintaining all intermediate states, depth-first search methods [1, 6, 24] have been proposed. DF-GED [1] uses a label set-based lower bound to prune inefficient paths in the search tree, while CSI-GED [24] employs a degree-based lower bound to enumerate edge mappings. However, these lower bounds are often loose, and while the performance of search-based methods is efficient for small graphs, it tends to deteriorate significantly on large graphs.

2.2.2 *Assignment-based methods.*

They reformulate GED as a quadratic assignment problem (QAP) [8], which can be addressed using techniques such as the integer projected fixed point algorithm [33] or the graduated non-convexity and concavity procedure [42]. However, QAP remains non-convex and computationally expensive, potentially requiring exponential time in the worst case. To mitigate this complexity, many methods further simplify the quadratic term by considering only the local structure of the graph, converting it to a linear sum assignment problem [31]. For example, Zeng et al. [82] approximates assignment costs by comparing only the 1-hop local star structure of each node, while Justice and Hero [29] propose a binary linear programming formulation of GED. While linearization reduces computational complexity, it may oversimplify the original problem and lead to suboptimal solutions. To address this, error correction techniques are often applied to refine the node mapping, typically involving enumerating and adjusting local node mappings to improve the overall solution quality. Several recent graph alignment approaches [40, 64] also leverage the ideas of assignment-based methods, but they focus more on establishing node correspondences across two attributed graphs rather than graph-level similarities.

2.2.3 *Learning-based methods.*

They leverage graph neural networks (GNNs) to encode graphs and perform GED computation.

These approaches can be broadly categorized into two groups: direct GED estimation and hybrid methods that assist classic algorithms. Direct GED estimation methods aim to learn and predict GED without relying on traditional search or assignment algorithms. SimGNN [4] employs a Siamese GNN architecture to jointly learn node embeddings and predict GED. TaGSim [3] introduces a type-aware framework that estimates GED by considering different graph edit types. Peng et al. [53] propose a GED-specific loss function that encodes multiple optimum node mappings while enforcing one-to-one constraints. Ghashing [56] trains a GNN to generate embeddings and hash codes that preserve GED between graphs using ground-truth GED results.

On the other hand, hybrid methods integrate learning-based approaches with classic algorithms to enhance GED computation. Noah [78] utilizes a graph path network to optimize the search direction of the A* algorithm. Wang et al. [72] incorporate a dynamic graph embedding network for similarity score prediction within the path search procedure. MATA* [39] combines learnable node matching with A* search for GED computation. GEDGNN trains GNNs to predict both GED values and node matching matrices, assisting in post-processing of edit paths. While these learning-based methods typically offer improved efficiency, they often lack optimality and feasibility guarantees, require substantial training data, and may struggle with generalization to unseen scenarios.

2.3 Fused Gromov-Wasserstein (FGW) Distance

Optimal transport aims to identify a transportation plan, represented by the coupling matrix π, that minimizes the cumulative cost of redistributing "mass" between two discrete probability distributions. For graphs g and h, we define uniform distributions μ and ν over their respective node sets $\boldsymbol{V}^g = \{u_i\}_{i=1}^n$ and $\boldsymbol{V}^h = \{v_j\}_{j=1}^n$ (including dummy nodes for equal size): $\mu = \frac{1}{n}\sum_{i=1}^n \delta_{u_i}$, $\nu = \frac{1}{n}\sum_{j=1}^n \delta_{v_j}$, where δ_{u_i} and δ_{v_j} are Dirac delta functions, assigning equal probability mass $\frac{1}{n}$ to each node.

The set of valid transportation plans $\Pi(\mu, \nu)$, denoted as Π_n for brevity, contains all joint distributions with marginals μ and ν:

$$\Pi(\mu, \nu) = \{\pi \geq 0 : \pi 1_n = \mu, \pi^T 1_n = \nu\} \tag{4}$$

where π_{ij} represents mass transported from node i to j, and 1_n is an n-dimensional ones vector. The coupling matrix $\pi \in \Pi_n$ can be viewed as a probabilistic node matching, generalizing permutation matrices by relaxing binary constraints to non-negativity.

The coupling matrix $\pi \in \Pi$ can be interpreted as a probabilistic matching between graph nodes. Any permutation matrix $\boldsymbol{P}$, when normalized by n (i.e., $\frac{\boldsymbol{P}}{n}$), represents a feasible solution within Π. Essentially, π generalizes permutation matrices by relaxing the binary assignment constraint to non-negativity, thereby significantly reducing the optimization difficulty.

Wasserstein Distance (WD) measures the distance between probability distributions μ and ν:

$$\mathrm{WD}(\boldsymbol{C}) := \min_{\pi \in \Pi_n} f_{\mathrm{WD}}(\boldsymbol{C}, \pi) = \min_{\pi \in \Pi_n} \sum_{i,j=1}^{n} \boldsymbol{C}_{ij}\pi_{ij} = \min_{\pi \in \Pi_n} \langle C, \pi \rangle \tag{5}$$

where $\boldsymbol{C}_{ij}$ represents the cost of transporting mass from u_i to v_j. For graph matching, $\boldsymbol{C}$ corresponds to the node relabeling cost matrix from Definition 2.2.

Gromov-Wasserstein Distance (GWD) extends classical OT to compare distributions in different metric spaces, making it particularly suitable for comparing graph structures.According to the work of Li et al. [35] by leveraging the l_2-norm variant, the GWD between the adjacency matrices $\boldsymbol{A}^g$ and $\boldsymbol{A}^h$ can be calculated as follows:

$$\begin{aligned} \mathrm{GWD}(\boldsymbol{A}^g, \boldsymbol{A}^h) &:= \min_{\pi \in \Pi_n} f_{\mathrm{GWD}}(\boldsymbol{A}^g, \boldsymbol{A}^h, \pi) \\ &= \min_{\pi \in \Pi_n} \sum_{i,j,k,l} |\boldsymbol{A}^g_{ij} - \boldsymbol{A}^h_{kl}|^2 \pi_{ik}\pi_{jl} \\ &= \min_{\pi \in \Pi_n} \|\boldsymbol{A}^g\|_F^2 + \|\boldsymbol{A}^h\|_F^2 - 2\mathrm{Tr}(\boldsymbol{A}^g \pi \boldsymbol{A}^h \pi^T) \end{aligned} \tag{6}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\mathrm{Tr}(\cdot)$ is the matrix trace. Higher values of π_{ik} and π_{jl} indicate stronger matching between node pairs (u_i, v_k) and (u_j, v_l), with the objective of minimizing structural differences $|\boldsymbol{A}^g_{ij} - \boldsymbol{A}^h_{kl}|$. For isomorphic graphs with correct node correspondence, GWD equals zero, indicating perfect structural alignment.

Fused Gromov-Wasserstein Distance (FGW) [66] combines WD and GWD to capture both node labels and structural information:

$$\begin{aligned} \mathrm{FGW}(\boldsymbol{C}, \boldsymbol{A}^g, \boldsymbol{A}^h, \alpha) &:= \min_{\pi \in \Pi_n} f_{\mathrm{FGW}}(\boldsymbol{C}, \boldsymbol{A}^g, \boldsymbol{A}^h, \alpha, \pi) \\ &= \min_{\pi \in \Pi_n} \alpha f_{\mathrm{WD}}(\boldsymbol{C}, \pi) + (1-\alpha) f_{\mathrm{GWD}}(\boldsymbol{A}^g, \boldsymbol{A}^h, \pi) \end{aligned} \tag{7}$$

where $\alpha \in [0, 1]$ balances the contribution of WD and GWD. This unified framework provides a comprehensive approach for comparing labeled or attributed graphs.

Several approaches have leveraged OT for alignment-based tasks [12]. OTEA [52] extends the TransE embedding framework by incorporating group-level OT-based losses for supervised entity alignment. [43] adapts the OT framework to address dangling entity detection during knowledge graph alignment. In the unsupervised domain, SLOTAlign [64] jointly optimizes structure learning and OT-based alignment, while FGWEA [65] employs the Fused Gromov-Wasserstein distance to better capture structural similarities between knowledge graphs. Building upon these advances in OT-based graph alignment, our work is the first to explore and optimize the FGW framework specifically for GED computation.

3 THE PROPOSED METHOD

This section presents FGWAlign, our novel framework for GED computation through enhanced FGW optimization. We begin by reformulating graph alignment-induced GED computation as an FGW problem in Section 3.1. To overcome limitations of conventional FGW solvers, we introduce three innovative strategies: diverse projection for escaping local optima in Section 3.2, random exploration for finding the global optimum in Section 3.3, and extension to multi-relational graphs in Section 3.4. Finally, we analyze the computational complexity of our proposed method in Section 3.5.

3.1 Reformulating GED to FGW

The key insight of our approach is that GED can be reformulated as an FGW problem through a specific choice of the fusion parameter α_0. This relationship is formalized in the following theorem:

THEOREM 3.1 (GED-FGW OBJECTIVE EQUIVALENCE). *For graphs with equal number of nodes $n = |\boldsymbol{V}^g| = |\boldsymbol{V}^h|$ and fusion parameter*

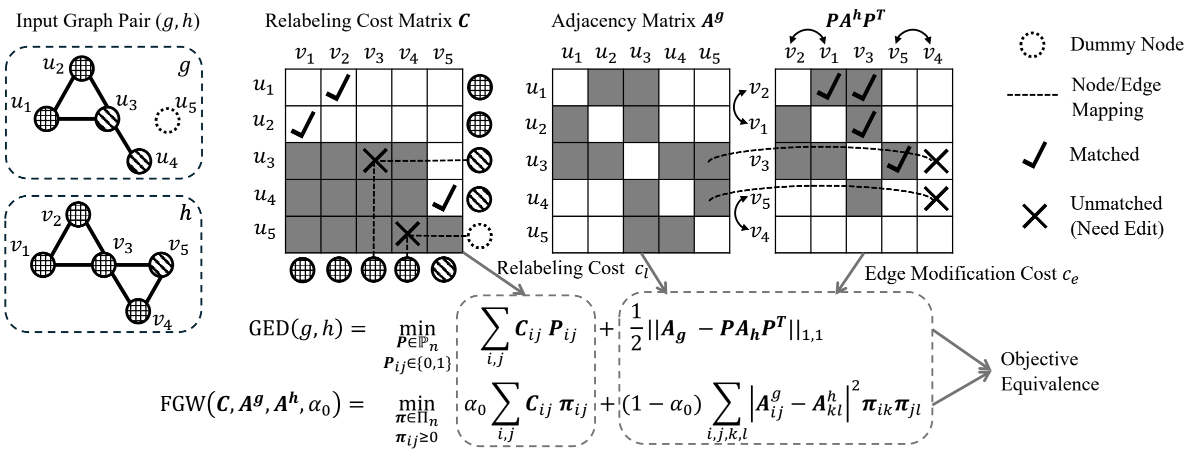

Input Graph Pair (g, h)
Relabeling Cost Matrix $\boldsymbol{C}$
Adjacency Matrix $\boldsymbol{A^g}$
$\boldsymbol{PA^hP^T}$
Dummy Node
Node/Edge Mapping
Matched
Unmatched (Need Edit)
Relabeling Cost c_l
Edge Modification Cost c_e
$\mathrm{GED}(g,h) = \min_{P \in \mathbb{P}_n, P_{ij} \in \{0,1\}} \sum_{i,j} C_{ij} P_{ij} + \frac{1}{2} \|A_g - PA_hP^T\|_{1,1}$
$\mathrm{FGW}(C, A^g, A^h, \alpha_0) = \min_{\pi \in \Pi_n, \pi_{ij} \geq 0} \alpha_0 \sum_{i,j} C_{ij} \pi_{ij} + (1-\alpha_0) \sum_{i,j,k,l} \left|A^g_{ij} - A^h_{kl}\right|^2 \pi_{ik}\pi_{jl}$
Objective Equivalence

Figure 2: Demonstrate of the GED calculation process for example 1 and the connection between GED and FGW.

```
Algorithm 1: Bregman Proximal Gradient descent for FGW
Input: 1. Adjacency matrices $A^g, A^h$, relabeling matrix $C$
       2. Initial plan $\pi_0$
       3. Step size $\eta$, number of iterations $N$, error bound $\epsilon$
Output: An updated transportation plan $\pi$
for $i = 1$ to $N$ do
    Compute the gradient matrix $\nabla_{\pi_{i-1}} f_{\mathrm{FGW}}$ by eq. (9)
    $\pi_i \leftarrow \arg\min_{\pi \in \Pi_n} \{\langle \nabla_{\pi_{i-1}} f_{\mathrm{FGW}}, \pi \rangle + \eta \mathbf{KL}(\pi || \pi^{i-1})\}$,
      solved by Sinkhorn algorithm;
    if $\|\pi_i - \pi_{i-1}\|_F < \epsilon$ then
        return $\pi_i$
return $\pi_N$
```

```
Algorithm 2: FGWAlign
Input: 1. Graph pair $g = \{V^g, E^g, l^g\}, h = \{V^h, E^h, l^h\}$
       2. Projection candidates $K$
       3. Exploration patience $T$
Output: The predicted GED value and corresponding
        assignment matrix.
Prepare a GED lower bound $d_{LB}(g, h)$
$\tau \leftarrow 0, d_{UB} \leftarrow \infty$
while $\tau < T$ do
    $\pi_0 \leftarrow \arg\min_{\pi \in \Pi_n} \langle M, \pi \rangle + \sum_{ij} \pi_{ij} \ln \pi_{ij}, M_{ij} \sim \mathcal{N}(0, 1)$
    $\pi^* \leftarrow \arg\min_{\pi \in \Pi_n} f_{\mathrm{FGW}}(A^g, A^h, C, \pi_0, \alpha_0)$
    $P_0 \leftarrow \arg\max_{P \in \mathbb{P}_n} \langle \pi^*, P \rangle$.
    for $k = 1$ to $K$ do
        $P_k = \arg\max_{P \in \mathbb{P}_n} \langle \pi^* - \lambda \Sigma_{i=0}^{k-1} P_i, P \rangle$
        $\hat{d} \leftarrow f_{\mathrm{GED}}(A^g, A^h, C, P_k)$
        if $\hat{d} < d_{UB}$ then
            $d_{UB} \leftarrow \hat{d}, P_{best} \leftarrow P_k, \tau \leftarrow 0$
        if $d_{UB} = d_{LB}$ then
            return $d_{UB}, P_{best}$
    $\tau \leftarrow \tau + 1$
return $d_{UB}, P_{best}$
```

$\alpha_0 = \frac{2}{n+2}$, *the GED and FGW objectives are related as:*

$$c_l = n f_{WD}(C, \frac{P}{n}), c_e = \frac{n^2}{2} f_{GWD}(A^g, A^h, \frac{P}{n}),$$
$$f_{GED}(C, A^g, A^h, P) = c_l + c_e = \frac{n}{\alpha_0} f_{FGW}(C, A^g, A^h, \alpha_0, \frac{P}{n}). \tag{8}$$

The complete proof of this equivalence is provided in Section 4.1. This theoretical relationship enables a practical solution: we can solve the discrete GED problem by first obtaining a continuous FGW solution π and then scaling it to derive an approximated discrete solution $P \approx n\pi$.

Figure 2 illustrates the calculation process using the example in Figure 1. The GED calculation seeks an optimal node mapping $\boldsymbol{P}$ between input graph pair (g, h) that minimizes two types of costs: node relabeling costs (encoded in matrix $\boldsymbol{C}$) and edge modification costs (derived from adjacency matrices $\boldsymbol{A}^g$ and $\boldsymbol{A}^h$). As shown in the example, when nodes u_3 and u_5 are mapped to v_3 and v_4 respectively, both cost types are incurred due to mismatched node labels and different structural connections marked with ×. The FGW formulation captures these same aspects through two corresponding terms: node relabeling costs $\alpha_0 \sum C_{ij}\pi_{ij}$ and structural differences $(1 - \alpha_0) \sum |A^g_{ij} - A^h_{kl}| \pi_{ik}\pi_{jl}$. When setting $\alpha_0 = \frac{2}{n+2}$, these objectives become equivalent up to a scaling factor, allowing us to convert an optimal FGW solution π into an optimal GED solution via $P \approx n\pi$.

We employ the Bregman Proximal Gradient descent (BPG) method to solve the resulting FGW problem [76], which has shown superior performance especially on large graphs and proved to have linear convergence ratio [34]. Algorithm 1 outlines the BPG process, which iteratively updates π using:

$$\nabla_\pi f_{\mathrm{FGW}}(C, A^g, A^h, \alpha_0, \pi) = \alpha_0 C + 4(1 - \alpha_0) A^g \pi A^h,$$
$$\pi_i = \arg\min_{\pi \in \Pi_n} \left\{ \langle \nabla_\pi f_{\mathrm{FGW}}, \pi \rangle + \eta \mathbf{KL}(\pi || \pi^{i-1}) \right\}. \tag{9}$$

The detailed derivation for this gradient form is presented in the appendix of our GitHub repository. The π-update is solved efficiently using the Sinkhorn algorithm [15]. We iteratively update π until either convergence is achieved or the maximum number of iterations N is reached. The resulting solution π^* is typically a continuous matrix containing more than n non-zero elements. We choose KL divergence as the regularization term because of its strong convexity, which transforms our non-convex optimization into more tractable convex subproblems. This regularization enforces that each iteration's transport plan π_i doesn't diverge drastically from the previous π_{i-1}, ensuring more stable convergence. Additionally, it creates an entropy barrier preventing degenerate solutions and promoting numerically stable transport plans. The Sinkhorn algorithm efficiently solves the KL-regularized subproblem through an alternating scaling procedure with $O(n^2)$ complexity per iteration—significantly faster than traditional $O(n^3 \log(n))$ linear programming approaches. Its matrix-vector operations are highly parallelizable for GPU acceleration, and it guarantees convergence to the unique optimal solution of the regularized subproblem.

3.2 Escaping Local Optimum via Diverse Projection

The projection from the transportation plan π^* to the assignment matrix $\boldsymbol{P}$ can be formulated as a maximum linear assignment problem, expressed as $\boldsymbol{P}_0 = \arg\max_{\boldsymbol{P} \in \mathbb{P}_n} \langle \pi^*, \boldsymbol{P} \rangle$. While traditional algorithms such as the Hungarian method or network simplex algorithms [7] can effectively solve this problem, their computational complexity of at least $O(|\boldsymbol{V}|^3)$ renders them inefficient for handling large graphs. To address this inefficiency, we propose a greedy approach that reduces the complexity to $O(|\boldsymbol{V}|^2)$.

Fast Greedy Assignment. The greedy method operates by applying a threshold to the elements of π^*. Specifically, we set all elements below $1/(n \log n)$ to zero. This step effectively filters out weak candidates, allowing us to focus on stronger elements. The remaining non-zero elements are then sorted in descending order. Subsequently, we assign these elements to $\boldsymbol{P}_0$ while ensuring that their corresponding rows and columns remain unassigned. In cases where rows or columns remain unassigned after this process, we randomly allocate the remaining zero entries. Although this approach may not yield the optimal assignment, it is particularly effective when π^* closely approximates the ideal assignment matrix.

Diverse Projection strategy. Building on the insights from prior studies [40, 55], we recognize that the objective function for the GED is highly non-convex. Additionally, the computational cost of verifying an assignment is relatively low. These properties allow us to explore multiple assignment configurations, thereby increasing our chances of escaping local optima. To achieve diverse solutions, we introduce a diverse projection strategy. For the k-th projection, we incorporate a penalty term into the assignment process, which ensures that the new assignment $\boldsymbol{P}_k$ differs from the previous $k-1$ projections. This is represented as follows:

$$\boldsymbol{P}_k = \arg\max_{\boldsymbol{P} \in \mathbb{P}_n} \left(\langle \pi^*, \boldsymbol{P} \rangle - \lambda \sum_{i=0}^{k-1} \langle \boldsymbol{P}_i, \boldsymbol{P} \rangle \right), \tag{10}$$

where λ is a hyperparameter that governs the degree of diversity among the projections. For simplicity, we fix $\lambda = 1/n$. In each iteration, the fast greedy assignment can be used.

The diverse projection strategy is introduced in lines 7-11 of Algorithm 2. During each projection, we calculate the current estimate of the GED value, denoted as $\hat{d}$, and compare it with the best-known estimate $\hat{d}_{UB}$.

3.3 Random Exploration of Global Optimum

While the diverse projection strategy enhances local search, it may not suffice to find the global optimum as it only explores the vicinity of the temporary BPG solution π^*. Moreover, the BPG algorithm for the FGW solver is sensitive to initialization and may converge to local optima, potentially limiting the effectiveness of diverse projection. To conduct a more comprehensive search of the solution space, we propose a random exploration approach that enhances the probability of identifying the global optimum.

As indicated in line 4 of Algorithm 2, we introduce an exploration patience parameter T to govern the number of exploration iterations. In each iteration, we sample a ground cost matrix $\boldsymbol{M}$ from a Gaussian distribution and use the Sinkhorn distance to derive a feasible initial point π_0. If we set $T = 1$ to discard random exploration, π_0 will be initialized as the uniform matrix, i.e., $\pi_0(i, j) = 1/n^2$. We then implement the diverse projection strategy to generate multiple diverse projections and obtain several candidate mappings. If these mappings yield a better estimation of the GED upper bound d_{UB}, we reset the patience counter τ to 0.

To accelerate the exploration process, we can precompute a GED lower bound $d_{LB}(g, h)$. If the current best estimation $\hat{d}_{UB}$ matches the lower bound, we can directly return the result. In our implementation, we utilize the label set lower bound [11] due to its computational efficiency. More sophisticated lower bounds could be employed to better guide the exploration process.

3.4 Extending to Multi-relational Graphs

Many real-world graphs feature edges with multiple types. For instance, atoms in a molecular graph are connected by three main types of chemical bonds. Unfortunately, FGW's fomulation are not natively support edge type modelling.

A straightforward approach to handling such multi-relational graphs is to treat each edge type as an independent graph and sum the cost. However, this method fails to adequately support edge modification operations. Changing the label of a specific edge type in this model would require removing an existing edge and adding a new one, which is suboptimal.

To address this limitation, we propose an approach inspired by the dummy node concept introduced in Section 2.1. We consider all unconnected node pairs as a new edge type, allowing us to unify edge addition or deletion operations into edge type modifications. Specifically, we can change an edge's label from its original type to this new *unconnected* type (to effectively remove the edge), or vice versa (to add an edge). Suppose we have R edge types in two graphs, represented by adjacency matrices $\{\boldsymbol{A}_i^g\}_1^r$ and $\{\boldsymbol{A}_i^h\}_1^r$. We first compute the new adjacency matrices of the *unconnected* type: $\boldsymbol{A}_{r+1}^g = \mathbf{1}_{n \times n} - \sum_{i=1}^r \boldsymbol{A}_i^g$ and $\boldsymbol{A}_{r+1}^h = \mathbf{1}_{n \times n} - \sum_{i=1}^r \boldsymbol{A}_i^h$. The GWD term in FGW can then be extended to multi-relational graphs as: $\mathrm{GWD}(g, h) = \min_{\pi \in \Pi_n} \sum_{i=1}^{r+1} \frac{1}{2} f_{\mathrm{GWD}}(A_i^g, A_i^h, \pi)$. The division by 2 accounts for each edge type modification operation contributing twice. This formulation naturally reduces to the case of graphs with a single edge type.

3.5 Complexity Analysis

The computational complexity of our proposed FGWAlign solver is primarily governed by the FGW optimization process. The BPG algorithm, as outlined in Algorithm 1, needs up to N iterations, where each iteration involves a complexity of $O(|\boldsymbol{V}||\boldsymbol{E}|)$ for gradient computation and $O(|\boldsymbol{V}|^2)$ for Sinkhorn iterations. Here, $|\boldsymbol{V}|$ and $|\boldsymbol{E}|$ represent the maximum count of nodes and edges in two graphs, respectively. Additionally, the diverse projection strategy entails K iterations, each requiring a complexity of $O(|\boldsymbol{V}|^3)$ for solving the linear assignment problem, but can be reduced to $O(|\boldsymbol{V}|^2)$ using our proposed greedy assignment approach. For a single attempt without random exploration, the total complexity of our FGWAlign solver is $O(N|\boldsymbol{V}||\boldsymbol{E}| + K|\boldsymbol{V}|^2)$ under normal conditions and $O(N|\boldsymbol{V}|^3 r)$ for multi-relational graphs. Given that the number of iterations N and the number of projection candidates K are generally set to small values, the time complexity for normal large-scale graphs is $O(|\boldsymbol{V}||\boldsymbol{E}|)$, and the space complexity is $O(|\boldsymbol{V}|^2)$.

4 THEORETICAL ANALYSIS

This section establishes the theoretical foundation of FGWAlign for GED computation. In Section 4.1, we prove the equivalence between FGW and GED objective functions. Section 4.2 analyzes the relationship between the optimal transportation plan π^* and the optimal assignment matrix $\boldsymbol{P}^*$. We first derive an error bound to quantify how this relationship affects GED estimation, then introduce a novel sparsity regularization term to the FGW formulation that ensures optimal value equivalence between FGW and GED.

4.1 Objective Function Equivalence

Proof of Theorem 3.1. Given that the entries of $\boldsymbol{A}^g$ and $\boldsymbol{A}^h$ are binary (0 or 1), and $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{A}^g_{ij} - (\boldsymbol{P}\boldsymbol{A}^h\boldsymbol{P}^T)_{ij}$ takes values in $\{-1, 0, 1\}$. Consequently, $|\boldsymbol{A}^g_{ij} - (\boldsymbol{P}\boldsymbol{A}^h\boldsymbol{P}^T)_{ij}| = |\boldsymbol{A}^g_{ij} - (\boldsymbol{P}\boldsymbol{A}^h\boldsymbol{P}^T)_{ij}|^2$. We have the edge modification cost:

$$\begin{aligned} c_e &= \frac{1}{2}\|\boldsymbol{A}^g - \boldsymbol{P}\boldsymbol{A}^h\boldsymbol{P}^\top\|_{1,1} = \frac{1}{2}\|\boldsymbol{A}^g - \boldsymbol{P}\boldsymbol{A}^h\boldsymbol{P}^\top\|_F^2 \\ &= \frac{1}{2}(\|\boldsymbol{A}^g\|_F^2 + \|\boldsymbol{A}^h\|_F^2 - 2\mathrm{Tr}(\boldsymbol{A}^g\boldsymbol{P}\boldsymbol{A}^h\boldsymbol{P}^\top)) \\ &= |\boldsymbol{E}^g| + |\boldsymbol{E}^h| - \mathrm{Tr}(\boldsymbol{A}^g\boldsymbol{P}\boldsymbol{A}^h\boldsymbol{P}^\top), \end{aligned} \tag{11}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\mathrm{Tr}(\cdot)$ is the matrix trace. On the other front,

$$\begin{aligned} f_{\mathrm{GWD}}(\boldsymbol{A}^g,\boldsymbol{A}^h,\frac{\boldsymbol{P}}{n}) &= \sum_{i,j,k,l}|\boldsymbol{A}^g_{ij} - \boldsymbol{A}^h_{kl}|^2\left(\frac{\boldsymbol{P}_{ij}}{n}\right)\left(\frac{\boldsymbol{P}_{kl}}{n}\right) \\ &= \sum_{i,k}\frac{|\boldsymbol{A}^g_{ik}|}{n^2} + \sum_{j,l}\frac{|\boldsymbol{A}^h_{jl}|}{n^2} - \frac{2}{n^2}\mathrm{Tr}(\boldsymbol{A}^g\boldsymbol{P}\boldsymbol{A}^h\boldsymbol{P}^T) \\ &= \frac{2}{n^2}\left(|\boldsymbol{E}^g| + |\boldsymbol{E}^h| - \mathrm{Tr}(\boldsymbol{A}^g\boldsymbol{P}\boldsymbol{A}^h\boldsymbol{P}^T)\right). \end{aligned} \tag{12}$$

Therefore, we establish that $c_e = \frac{n^2}{2}f_{\mathrm{GWD}}(\boldsymbol{A}^g, \boldsymbol{A}^h, \frac{\boldsymbol{P}}{n})$, which derives the objective equivalence between GED and FGW:

$$\begin{aligned} f_{\mathrm{GED}}(\boldsymbol{C}, \boldsymbol{A}^g, \boldsymbol{A}^h, \boldsymbol{P}) &= \langle \boldsymbol{C}, \boldsymbol{P}^*\rangle + \frac{1}{2}\|\boldsymbol{A}^g - \boldsymbol{P}^*\boldsymbol{A}^h\boldsymbol{P}^{*T}\|_{1,1} \\ &= n\langle \boldsymbol{C}, \frac{\boldsymbol{P}}{n}\rangle + \frac{n^2}{2}f_{\mathrm{GWD}}(\boldsymbol{A}^g, \boldsymbol{A}^h, \frac{\boldsymbol{P}}{n}) \\ &= \frac{n}{\alpha_0}\left(\alpha_0 + (1-\alpha_0)f_{\mathrm{GWD}}(\boldsymbol{A}^g, \boldsymbol{A}^h, \frac{\boldsymbol{P}}{n})\right) \\ &= \frac{n}{\alpha_0}f_{\mathrm{FGW}}(\boldsymbol{C}, \boldsymbol{A}^g, \boldsymbol{A}^h, \alpha_0, \boldsymbol{P}/n) \end{aligned} \tag{13}$$

This completes the proof. □

4.2 Optimal Value Equivalence and Error Bound

Although the objective functions of FGW and GED can be equivalent, the optimal solution of FGW may not correspond to that of GED. This discrepancy arises from differences in their feasible sets. Specifically, after scaling the transportation plan π by n, FGW optimizes over doubly stochastic matrices ($\mathbb{D}_n$), which constitute the convex hull of permutation matrices ($\mathbb{P}_n$).

In this section, we bridge the gap between $\mathbb{D}_n$ and $\mathbb{P}_n$ by introducing a novel sparsity regularization term to the FGW formulation, promoting sparse solutions. Based on this analysis, we then derive an error bound for GED estimation using FGWAlign. For simpliciy, we define $\hat{f}_{\mathrm{FGW}}(\boldsymbol{D}) = \frac{n}{\alpha_0}f_{\mathrm{FGW}}(\boldsymbol{C}, \boldsymbol{A}^g, \boldsymbol{A}^h, \alpha_0, \boldsymbol{D}/n)$, where $\boldsymbol{D} \in \mathbb{D}$ is in the set of the doubly stochastic matrix (i.e., the space of $n\pi$). we first analyze the Lipschitz property of this function, which plays a crucial role in the subsequent proof.

Lemma 4.1 (Lipshcitz Property). *$\hat{f}_{\mathrm{FGW}}(\boldsymbol{D})$ is L-Lipschitz on the set of doubly stochastic matrices $\mathbb{D}_n$, where $L = \|\boldsymbol{C}\|_F + n^2$.*

Proof. The gradient of $\hat{f}_{\mathrm{FGW}}$ is given by

$$\nabla\hat{f}_{\mathrm{FGW}}(\boldsymbol{D}) = \boldsymbol{C} + \left(\sum_{j,l}|\boldsymbol{A}^g_{ij} - \boldsymbol{A}^h_{kl}|^2\boldsymbol{D}_{jl}\right)_{ik}.$$

Since $|\boldsymbol{A}^g_{ij} - \boldsymbol{A}^h_{kl}| \le 1$ and $\sum_{j,l}\boldsymbol{D}_{jl} = n$, we have $\|(\sum_{j,l}|\boldsymbol{A}^g_{ij} - \boldsymbol{A}^h_{kl}|^2\boldsymbol{D}_{jl})_{ik}\|_F \le n^2$. Hence, $\|\nabla\hat{f}_{\mathrm{FGW}}(\boldsymbol{D})\|_F \le \|\boldsymbol{C}\|_F + n^2$ for $\boldsymbol{D} \in \mathbb{D}_n$. By the mean value theorem, we complete the proof. □

Then, we introduce a lemma from Liu et al. [38], which establishes an upper bound for $\mathrm{dist}(\boldsymbol{P}, \mathbb{P}_n)$. This upper bound will serve as the penalty term in our analysis.

Lemma 4.2 (Error bound for permutation matrices [38]). *For any doubly stochastic matrices $\boldsymbol{D} \in \mathbb{D}_n$, we have the error bound*

$$\mathrm{dist}(\boldsymbol{D}, \mathbb{P}_n) \le 3\sqrt{n}\left(n - \|\boldsymbol{D}\|_F^2\right). \tag{14}$$

With Lemma 4.1 and 4.2, we are now ready to prove the following exact penalty theorem, which establishes the connection between the optimum value of $\hat{f}_{\mathrm{FGW}}$ and $\mathrm{GED}(g, h)$.

Theorem 4.3 (Optimal Value Equivalence). *When $\lambda \ge 3\sqrt{n}L$,*

$$\mathrm{GED}(g, h) = \min_{\boldsymbol{P}\in\mathbb{P}_n} f_{\mathrm{GED}}(\boldsymbol{P}) = \min_{\boldsymbol{P}\in\mathbb{D}_n}\hat{f}_{\mathrm{FGW}}(\boldsymbol{D}) + \lambda(n - \|\boldsymbol{D}\|_F^2).$$

Proof. Let $\boldsymbol{P}^*$ be the optimal solution to GED. Note that $\boldsymbol{P}^* \in \mathbb{P}_n$, then we have $n - \|\boldsymbol{P}^*\|_F^2 = 0$. Thus,

$$\mathrm{GED}(g, h) = f_{\mathrm{GED}}(\boldsymbol{P}^*) = \hat{f}_{\mathrm{FGW}}(\boldsymbol{P}^*) \ge \min_{\boldsymbol{D}\in\mathbb{D}_n}\hat{f}_{\mathrm{FGW}}(\boldsymbol{D}) + \lambda(n - \|\boldsymbol{D}\|_F^2).$$

For the other direction, as $\hat{f}_{\mathrm{FGW}}$ is Lipschitz continuous on $\mathbb{D}_n$ with Lipschitz constant L, we have:

$$\begin{aligned} &\forall \boldsymbol{D}_1, \boldsymbol{D}_2,\ \hat{f}_{\mathrm{FGW}}(\boldsymbol{D}_2) - \hat{f}_{\mathrm{FGW}}(\boldsymbol{D}_1) \le L\|\boldsymbol{D}_2 - \boldsymbol{D}_1\|_F, \\ \Rightarrow &\forall \boldsymbol{D}_1,\ \min_{\boldsymbol{D}_2\in\mathbb{P}_n}\hat{f}_{\mathrm{FGW}}(\boldsymbol{D}_2) - \hat{f}_{\mathrm{FGW}}(\boldsymbol{D}_1) \le L\min_{\boldsymbol{D}_2\in\mathbb{P}_n}\|\boldsymbol{D}_2 - \boldsymbol{D}_1\|_F, \\ \Rightarrow &\forall \boldsymbol{D}_1,\ \mathrm{GED}(g, h) \le \hat{f}_{\mathrm{FGW}}(\boldsymbol{D}_1) + L\mathrm{dist}(\boldsymbol{D}_1, \mathbb{P}_n). \end{aligned}$$

By Lemma 4.2, as $\lambda \geq 3\sqrt{n}L$,

$$\lambda(n - \|P\|_F^2) \geq \frac{\lambda}{3\sqrt{n}} \mathrm{dist}(P, \mathbb{P}_n) \geq L\mathrm{dist}(D_1, \mathbb{P}_n)$$

$$\mathrm{GED}(g,h) \leq \min_{D \in \mathbb{D}_n} \hat{f}_{\mathrm{FGW}}(D) + \lambda(n - \|D\|_F^2).$$

This completes the proof. □

Theorem 4.3 establishes an equivalence between discrete optimization over permutation matrices and continuous optimization over doubly stochastic matrices through an exact penalty formulation. However, the required large penalty term λ poses numerical stability and computational challenges. Therefore, instead of using this penalty term, FGWAlign employs diverse projection and random exploration strategies to search for multiple candidate solutions and increase the likelihood of finding the global optimum. This proof leads to the following error bound for GED estimation:

Corollary 4.4 (Error Bound for GED Estimation). *Let D^* be the optimal solution to FGW, and $\hat{P}$ be its corresponding projected permutation matrix. The difference between the optimal GED value $f_{\mathrm{GED}}(P^*)$ and the estimated GED value $f_{\mathrm{GED}}(\hat{P})$ is bounded by:*

$$|f_{\mathrm{GED}}(\hat{P}) - f_{\mathrm{GED}}(P^*)| \leq \hat{f}_{\mathrm{FGW}}(D^*) + L\mathrm{dist}(D^*, \hat{P}) - \hat{f}_{\mathrm{FGW}}(D^*)$$
$$\leq (\|C\|_F + n^2)\mathrm{dist}(D^*, \hat{P}).$$

This bound achieves tightness only when $\mathrm{dist}(D^*, \hat{P})$ is small, i.e., D^* is close to a permutation matrix. Such a condition typically holds for graph pairs that exhibit unique optimal solutions. For general graphs, establishing a non-trivial bound remains fundamentally challenging due to the NP hardness of GED computation.

5 EXPERIMENTS

5.1 Experimental Setup

5.1.1 Datasets and GED Pair Collection. Following prior studies [3, 4, 55], we evaluate on 5 graph datasets summarized in Table 3:

- **AIDS** [11, 87] comprises 42,869 antivirus screen chemical compounds. Nodes represent 29 types of atoms (labeled with elements like C, N, O, Cu), and edges represent three types of covalent bonds. We use the same test pairs as [3], consisting of 20 reference graphs paired with 800 query graphs.
- **AIDS700** [4, 55] samples 700 graphs in AIDS with no more than 10 nodes each, with node and edge labels removed. Following [55], we split the dataset into 6:2:2 for training, validation, and testing. For each test graph, 100 query graphs are randomly sampled from the training set, resulting in 140×100 test pairs.
- **Linux** [4, 73] collects 1,000 program dependence graphs derived from Linux kernel functions. Each graph corresponds to a function, with nodes representing statements and edges denoting dependencies. Test pairs are generated similarly to the AIDS dataset, forming 200×100 test pairs. Ground-truth GED values in AIDS, AIDS700, and Linux are computed using exact search.
- **IMDB** [3, 77] consists of 1,500 relatively large unlabeled graphs, where nodes represent actors or actresses and edges represent co-starring relationships. For graphs <10 nodes, exact GED is computed. For larger graphs, following [55], we generate 100 synthetic variants per test graph.
- **Synthetic**: Our approach to generating synthetic graphs is more diverse than previous studies [55, 78]. We combine power-law, Erdős-Rényi, Barabási-Albert, and Gaussian random partition graphs. Node counts are uniformly sampled from $\{2^k\}_{5 \leq k \leq 14}$, resulting in graphs with up to 16,384 nodes. We generate 100 graphs for each node count, totaling 1,000 graphs. The average edge count is maintained between 5 and 20 times the node count. For each graph g, we create two test pairs: (g, g_1) and (g, g_2). g_1 is derived by removing p% of nodes and their connected edges from g, while g_2 is created by adding p% new edges and altering p% of node labels in g. The ground-truth GED values are the number of operations performed. We set p to 25, allowing for GED values up to 40,000 between large graph pairs. This presents a more challenging scenario compared to previous synthetic datasets, which typically limited edit operations to no more than 20.

5.1.2 Competitors and Implementation Details. We compare our proposed FGWAlign with a comprehensive set of state-of-the-art GED solvers in three categories. All experiments are conducted on a high-performance Linux server with an AMD EPYC 7763 64-Core CPU and an NVIDIA RTX 4090 GPU. We implement FGWALign in Python 3.10 with PyTorch 2.0.1 and CUDA 11.7 for GPU acceleration (synthetic dataset only). We provide a brief overview of all baselines and implementation details below:

- **Search-based solvers**: We include three approximate beam-search-based solvers: A*-Hungarian [59], A*-Star [88], and A*-LSa [11]. Each solver utilizes a different heuristic to estimate the lower bound: the Hungarian algorithm [2], the star distance [82], and the label set heuristic, respectively. The implementations of these solvers are sourced from the Noah-GED repository[1] and the *pygmtools* library[2] [71]. The beam size for these algorithms is set to the maximum value from the set $\{1, 5, 10, 50, 100, 500, 1000\}$ that does not exceed a one-day runtime limit for each dataset. For example, the beam size for A*-LSa is set to 100 for the AIDS700 and Linux datasets, 5 for IMDB, and 1 for the Synthetic dataset.
- **Assignment-based algorithms**: We include IPFP [33], an integer projected fixed point method for graph alignment; RRWM [14], a reweighted random walk method for graph matching; SM [32], a spectral method for correspondence problems using pairwise constraints; and VJ [17, 28], a shortest augmenting path algorithm for linear assignment. We use the implementations in the *pygmtools* library with default hyperparameters.
- **Learning-based methods**: We consider five graph neural networks: two that directly predict GED values (SimGNN [4] and TaGSim [3]), and three that generate edit paths to compute GED (GEDGNN-m [55], Noah [78], and GEDHOT [13]). For all models, we utilize their publicly available source code and employ default configurations for both training and evaluation processes.
- **FGWAlign**: We evaluate five variants of FGW-based alignment: FGWAlign$_{\mathrm{full}}$ uses maximum candidates $K = 10$ in diverse projection and maximum patience $T = 20$; FGWAlign$_{\mathrm{rel}}$ considers edge relations with the same parameters ($K = 10$, $T = 20$); FGWAlign$_{\mathrm{fast}}$ uses $K = 10$ but reduced patience $T = 1$. For these three variants, we use step size $\beta = 0.1$ and maximum 200 epochs

[1]https://github.com/pkumod/Noah-GED/blob/main/src/ged.py
[2]https://pygmtools.readthedocs.io/en/latest/api/_autosummary/pygmtools.classic_solvers.html

Dataset	#Graphs	#Test Pairs	*avg.* $\lvert V\rvert$	*avg.* $\lvert E\rvert$	*max.* $\lvert V\rvert$	*max.* $\lvert E\rvert$	$\lvert\Sigma_v\rvert$	$\lvert\Sigma_e\rvert$	*avg.* d_{gt}	*max.* d_{gt}
AIDS	42,689	16,000	25.6	27.5	222	247	66	3	10.4	20
AIDS700	700	14,000	8.9	8.8	10	14	29	1	9.0	22
Linux	1,000	20,000	7.6	6.9	10	13	1	1	4.7	16
IMDB	1,500	30,000	13.0	65.9	89	1,467	1	1	7.1	36
Synthetic	1,000	2,000	3,274	23,211	16,384	245,264	5	1	7,144	90,998

Table 3: Statistics of the datasets, including the number of graphs (#Graphs) and test pairs (#Test Pairs), average and maximum numbers of nodes (*avg.* $\lvert V\rvert$, *max.* $\lvert V\rvert$) and edges (*avg.* $\lvert E\rvert$, *max.* $\lvert E\rvert$), sizes of node label set ($\lvert\Sigma_v\rvert$) and edge type set ($\lvert\Sigma_e\rvert$), as well as the average and maximum ground-truth GED values (*avg.* d_{gt} and *max.* d_{gt}).

Dataset	Method	Computation Quality				Similarity Ranking				Edit Path Accuracy			Efficiency
		Acc.	Fea.	RMSE	MAE	ρ	τ	p@10	p@20	R	P	F_1	Time(s/100p)
AIDS700	A*-Hungarian	0.621	0.990	1.038	0.574	0.901	0.831	0.889	0.906	0.764	0.741	0.752	98.982
	A*-Star	0.202	1.000	2.890	2.315	0.734	0.619	0.764	0.749	0.617	0.526	0.565	119.317
	A*-LSa	0.497	1.000	2.071	1.341	0.794	0.724	0.754	0.876	0.770	0.664	0.703	107.499
	IPFP	0.075	1.000	7.837	6.694	0.477	0.373	0.554	0.592	0.602	0.363	0.445	11.069
	RRWM	0.232	1.000	4.982	3.685	0.752	0.626	0.820	0.806	0.639	0.499	0.553	140.703
	Spectral	0.042	1.000	7.489	6.690	0.446	0.349	0.529	0.578	0.604	0.357	0.442	0.426
	VJ	0.020	1.000	8.243	7.569	0.466	0.365	0.483	0.570	0.557	0.311	0.393	0.175
	SimGNN	0.338	0.676	1.137	0.914	0.832	0.693	0.624	0.720	-	-	-	0.376
	TaGSim	0.366	0.662	1.105	0.841	0.850	0.715	0.646	0.746	-	-	-	0.152
	GEDGNN-m	0.433	1.000	2.403	1.528	0.796	0.692	0.843	0.843	0.722	0.652	0.681	52.002
	Noah	0.177	1.000	3.251	2.605	0.669	0.552	0.626	0.698	0.598	0.488	0.532	186.152
	GEDHOT	0.712	1.000	-	0.440	0.923	0.864	0.951	0.935	0.809	0.786	0.796	112.161
	FGWLibSolver	0.211	1.000	2.283	2.930	0.744	0.625	0.710	0.746	0.598	0.508	0.546	1.925
	FGWAlign$_{\text{light}}$	0.397	1.000	2.437	1.614	0.747	0.640	0.792	0.794	0.657	0.572	0.607	0.352
	FGWAlign$_{\text{fast}}$	0.609	1.000	1.190	0.662	0.884	0.807	0.896	0.891	0.721	0.685	0.701	1.718
	FGWAlign$_{\text{full}}$	**0.951**	1.000	**0.278**	**0.058**	**0.988**	**0.978**	**0.991**	**0.990**	**0.921**	**0.919**	**0.920**	49.809
Linux	A*-Hungarian	0.868	1.000	0.819	0.272	0.905	0.868	0.920	0.928	0.890	0.857	0.869	76.571
	A*-Star	0.440	1.000	2.315	1.593	0.843	0.765	0.920	0.938	0.792	0.670	0.716	66.574
	A*-LSa	0.497	1.000	2.071	1.341	0.794	0.724	0.754	0.876	0.770	0.664	0.703	107.499
	IPFP	0.007	1.000	6.774	6.364	0.623	0.544	0.637	0.746	0.774	0.339	0.459	9.037
	RRWM	0.038	1.000	6.501	5.867	0.658	0.570	0.530	0.684	0.715	0.353	0.456	5.499
	Spectral	0.071	1.000	6.405	5.666	0.524	0.445	0.668	0.664	0.769	0.398	0.502	0.379
	VJ	0.211	1.000	3.367	2.668	0.744	0.667	0.736	0.813	0.749	0.531	0.607	0.135
	SimGNN	0.596	0.800	0.621	0.456	0.933	0.844	0.891	0.920	-	-	-	0.378
	TaGSim	0.668	0.828	0.546	0.391	0.924	0.837	0.816	0.878	-	-	-	0.267
	GEDGNN-m	0.922	1.000	0.782	0.201	0.964	0.944	0.971	0.978	0.920	0.903	0.910	21.270
	Noah	0.616	1.000	1.877	1.062	0.796	0.726	0.808	0.879	0.826	0.738	0.770	55.841
	GEDHOT	0.984	1.000	-	0.033	0.994	0.990	0.992	0.996	0.928	0.924	0.926	47.523
	FGWLibSolver	0.453	1.000	1.376	2.017	0.853	0.775	0.842	0.864	0.738	0.624	0.666	4.714
	FGWAlign$_{\text{light}}$	0.606	1.000	1.997	1.163	0.858	0.788	0.934	0.922	0.843	0.734	0.774	0.304
	FGWAlign$_{\text{fast}}$	0.804	1.000	0.974	0.419	0.936	0.897	0.964	0.959	0.882	0.831	0.850	1.352
	FGWAlign$_{\text{full}}$	**0.997**	1.000	**0.104**	**0.005**	**0.999**	**0.998**	**0.998**	**0.999**	**0.930**	**0.930**	**0.930**	19.945
IMDB	A*-Hungarian	0.554	1.000	58.311	20.122	0.595	0.565	0.739	0.729	0.741	0.612	0.628	6.227
	A*-Star	0.705	1.000	17.347	5.963	0.807	0.771	0.871	0.868	0.725	0.598	0.620	109.630
	A*-LSa	0.600	1.000	43.815	15.684	0.631	0.605	0.743	0.738	0.758	0.597	0.618	70.419
	IPFP	0.872	1.000	3.322	0.741	0.949	0.932	0.975	0.970	**0.952**	0.908	0.920	14.893
	RRWM	0.872	1.000	9.794	2.425	0.939	0.916	0.957	0.957	0.944	0.874	0.886	4.576
	Spectral	0.736	1.000	18.829	7.156	0.842	0.804	0.868	0.870	0.885	0.746	0.765	0.466
	VJ	0.490	1.000	92.422	37.078	0.484	0.470	0.596	0.632	0.758	0.553	0.572	0.766
	SimGNN	0.178	0.423	3.820	2.579	0.349	0.326	0.620	0.656	-	-	-	0.379
	TaGSim	0.195	0.497	5.540	3.333	0.501	0.462	0.648	0.681	-	-	-	0.149
	GEDGNN-m	0.741	0.999	11.149	4.209	0.875	0.838	0.930	0.920	0.851	0.735	0.759	116.428
	Noah	0.573	1.000	45.627	14.029	0.644	0.614	0.765	0.771	0.770	0.612	0.637	64.958
	GEDHOT	0.950	1.000	-	0.254	0.983	0.972	0.995	0.993	0.946	0.927	0.933	170.412
	FGWLibSolver	0.825	1.000	1.081	3.533	0.887	0.856	0.972	0.948	0.887	0.834	0.850	3.194
	FGWAlign$_{\text{light}}$	0.951	1.000	4.270	0.658	0.959	0.949	0.972	0.971	0.923	0.904	0.908	0.270
	FGWAlign$_{\text{fast}}$	0.968	1.000	2.627	0.319	0.977	0.969	0.986	0.986	0.920	0.912	0.912	1.747
	FGWAlign$_{\text{full}}$	**0.996**	1.000	**0.370**	**0.022**	**0.997**	**0.995**	**0.999**	**0.998**	0.926	0.926	0.926	16.105

Table 4: Performance comparison of GED computation methods on AIDS700, Linux, and IMDB datasets in terms of computation quality, similarity ranking performance, edit path quality, and computational efficiency. Best results are bolded.

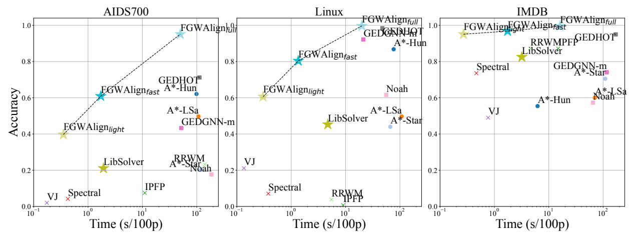


Figure 3: Comparison of computational efficiency and accuracy for various GED computation methods across AIDS700, Linux, and IMDB. The x-axis shows computation time (per 100 pairs) on a logarithmic scale, while the y-axis is accuracy.

for graphs under 100 nodes, and $\beta = 0.01$ with maximum 1000 epochs for larger graphs. FGWAlign$_{\text{light}}$ is a speed-optimized variant with $K = 1$, $T = 1$, and maximum 20 epochs. We also compare a standard FGW solver (named as FGWLibSolver) from the Python Optimal Transport library [19].

5.1.3 *Evaluation Metrics.* In line with prior works [4, 55, 78], we employ a comprehensive set of metrics across four categories to thoroughly assess our model's performance versus baselines:

- **GED Computation Quality.** We evaluate the computation quality of GED using four metrics: accuracy (**Acc.**), feasibility (**Fea.**), Mean Absolute Error (**MAE**), and Root Mean Squared Error (**RMSE**). Accuracy is the proportion of test pairs that satisfy $d_{pred} = d_{gt}$, where d_{pred} and d_{gt} are the predicted and ground-truth GED values, respectively. Feasibility is the proportion of test pairs where $d_{pred} \geq d_{gt}$. MAE is the average of $|d_{pred} - d_{gt}|$, and RMSE is the square root of the average of $|d_{pred} - d_{gt}|^2$ across all test pairs.
- **Similarity Ranking Performance.** For each test graph g, we randomly select 100 graphs to construct test pairs. These pairs are ranked based on their predicted or real GED to g. We evaluate the predicted rankings using Spearman's Rank Correlation Coefficient (ρ), Kendall's Rank Correlation Coefficient (τ), and precision at the top 10 and top 20 (p@10 and p@20).
- **Edit Path Quality.** We compare generated edit paths with ground truth using recall (R), precision (P), and F1-score (F_1).
- **Computational Efficiency.** For the real-world datasets, we record the average processing time per 100 graph pair comparisons. On the synthetic dataset, we measure the average computation time and peak memory usage per graph pair comparison.

5.2 Main Results

In Table 4 and Figure 3, we present the comparison beteen our propoased FGWAlign and 12 GED computation methods on AIDS700, Linux, and IMDB datasets. The results are analyzed as follows:

GED Computation Quality. FGWAlign$_{\text{full}}$ achieves the highest accuracy across all three datasets, with rates of 95.1%, 99.7%, and 99.6% for AIDS700, Linux, and IMDB respectively. This represents an 5-fold to 10-fold decrease in error rates compared to previous state-of-the-art methods. Additionally, FGWAlign$_{\text{full}}$ demonstrates the lowest RMSE and MAE across all datasets, underscoring its superior precision in GED estimation. The MAE values are decreased to 0.058, 0.005, and 0.022 on the three datasets respectively. In terms of feasibility, since FGWAlign can generate edit paths based on the assignment matrix, it serves as an upper bound for GED and guarantees 100% feasibility.

Time Efficiency. As demonstrated in Table 4, FGWAlign$_{\text{fast}}$ exhibits superior computational efficiency, processing 100 graph pairs within 3 seconds across all datasets. While Spectral and VJ methods demonstrate faster processing times, their accuracy falls significantly short. SimGNN and TaGSim offer faster computation but they only predict GED values without generating edit paths, providing incomplete solutions lacking feasibility guarantees. Moreover, their reported efficiency metrics exclude the necessary training time. Compared to state-of-the-art learning-based methods that generate edit paths, FGWAlign$_{\text{fast}}$ achieves a 15-60× speedup while maintaining comparable accuracy. Figure 3 illustrates that FGWAlign$_{\text{fast}}$'s accuracy-time trade-off curve decisively outperforms other methods, validating the effectiveness of our algorithmic improvements in optimizing both accuracy and computational efficiency.

Similarity Ranking. FGWAlign$_{\text{full}}$ demonstrates near-optimal performance in ranking graph similarities, achieving the highest scores across all four metrics in three datasets. For instance, FGWAlign$_{\text{full}}$ achieves a p@10 of 0.991 on the AIDS700 dataset, significantly outperforming the best baseline score of 0.889. These results showcase that FGWAlign is exceptionally effective in preserving the relative similarities between graphs, an essential aspect for various graph analysis tasks including retrieval, clustering, and classification.

Edit Path Accuracy. FGWAlign$_{\text{full}}$ achieves the highest precision and F1-scores across all datasets, with a particularly significant improvement on the AIDS700 dataset—where its score rises from 0.752 (the previous second-best) to 0.920. These results demonstrate FGWAlign$_{\text{full}}$'s ability to generate high-quality edit paths that precisely capture the optimal edit operations between graphs.

Impact of Dataset Characteristics on Performance. The experimental results reveal significant patterns in how graph properties influence method performance across datasets. On AIDS700, with smaller graphs (avg. $|V| = 8.9$) but complex node labeling ($|\Sigma_v| = 29$), traditional search-based methods like A*-Hungarian achieve moderate accuracy, while assignment-based methods have poor performance on this dataset. The Linux dataset, featuring unlabeled simple graphs with low average GED values (avg.$d_{gt} = 4.7$), shows improved performance across most methods, but assignment-based methods still perform poorly. Notably, on the IMDB dataset with larger, denser graphs (avg.$|E| = 65.9$), assignment-based methods perform surprisingly well (both IPFP and RRWM achieve 87.2% accuracy), while learning-based methods like SimGNN and TaGSim struggle significantly (17.8% and 19.5% accuracy). This suggests that dense, unlabeled graphs particularly challenge embedding-based learning approaches. Interestingly, assignment-based methods excel in edit path quality on IMDB (IPFP achieves the highest recall at 95.2%), but FGWAlign$_{\text{full}}$ still maintains superior overall performance with 99.6% accuracy and minimal error (MAE = 0.022). The consistent excellence of FGWAlign across datasets with varying structural complexities demonstrates its robust adaptability to different graph characteristics, effectively balancing structural and feature similarities in various settings.

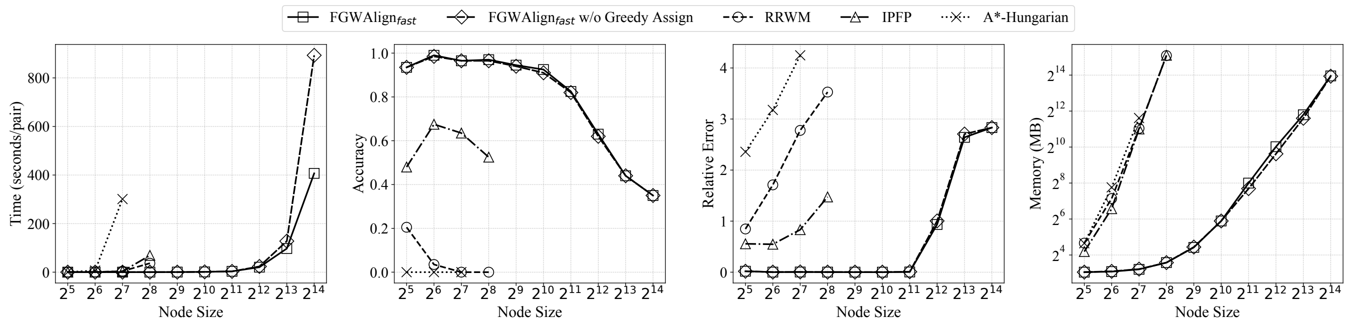


Figure 4: Scalability comparison between FGWAlign$_{\text{fast}}$ (with and without fast greedy assignment) and three baseline methods (RRWM, IPFP, and A*-Hungarian) on the synthetic dataset with increasing graph sizes.

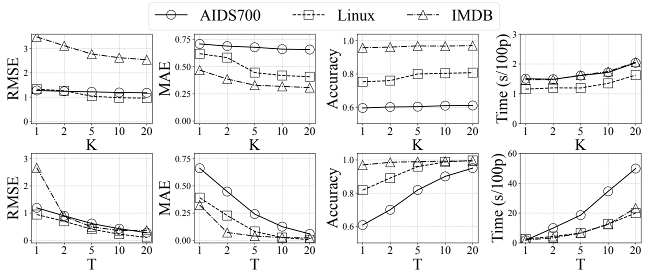


Figure 5: Impact of projection candidates K for FGWAlign$_{\text{fast}}$ and exploration patience T for FGWAlign$_{\text{full}}$.

5.3 Scalability Evaluation

To evaluate the scalability of our proposed FGWAlign with increasing graph scale, we conduct experiments on our synthetic dataset with graph sizes ranging from 2^5 to 2^{14} nodes. We compare FGWAlign$_{\text{fast}}$ against three baseline methods: RRWM, IPFP, and A*-Hungarian in terms of computation time, accuracy, the relative error between predicted GED and ground truth GED ($|d_{pred} - d_{gt}|/|d_{gt}|$), and peak memory usage. We also evaluate a specific variant of FGWAlign$_{\text{fast}}$ without greedy assignment, which instead uses linear assignment and the network simplex algorithm as a replacement.

Results in Figure 4 show FGWAlign$_{\text{fast}}$'s time and memory scale quadratically with node size, matching theoretical complexity. The variant without greedy assignment shows increased time costs for graphs over 5,000 nodes, while maintaining similar performance metrics, confirming our greedy assignment strategy's effectiveness. In contrast, A*-Hungarian's computation requirements surge between 64-128 nodes and fails at 256 nodes due to time constraints. RRWM and IPFP cannot process graphs beyond 512 nodes within 48 hours. For accuracy, FGWAlign$_{\text{fast}}$ maintains >0.8 accuracy with near-zero relative error for graphs under 2,048 nodes. Even with 16,384-node graphs (40× larger than previous works [55]), FGWAlign achieves >30% accuracy. Baseline methods struggle with scale: A*-Hungarian fails at just 32 nodes, while RRWM and IPFP show significant accuracy degradation as graph size increases, highlighting FGWAlign's superior scalability and efficiency.

FGWAlign faces scalability issues when handling million-scale graphs, primarily due to its quadratic complexity. In such cases, memory requirements would scale to an infeasible ∼50TB. To tackle this limitation, two promising directions can be explored: (1) adapting FGWAlign to operate within divide-and-conquer entity alignment frameworks [75, 81], which partition large graphs into manageable subgraphs; and (2) restricting the coupling matrix π to a sparse format (e.g., considering only top-k candidates per node) and leveraging techniques such as masked Optimal Transport [23], which could drastically reduce computational overhead.

5.4 Ablation Study

We evaluate three key enhancements in FGWAlign compared to the original FGW solver: diverse projection, random exploration, and multi-relational extension.

Impact of projection candidates K. Varying K from 1 to 20 in FGWAlign$_{\text{fast}}$, Figure 5 (top) shows monotonic quality improvements across all datasets with increasing K, particularly for IMDB and Linux where MAE decreases from 0.469 to 0.308 and 0.620 to 0.408, respectively. The computational overhead remains modest (only 15% additional time for $K = 10$), demonstrating the efficiency of K-best projection.

Impact of exploration patience T. Figure 5 shows significant quality improvements as T increases from 1 to 20, with IMDB showing 67% RMSE reduction when T increases from 1 to 2. However, this comes with substantial computational costs (30x longer for AIDS, 10x for Linux/IMDB). We recommend balancing large K with appropriate T for optimal quality-efficiency trade-offs.

Effectiveness of edge type modeling. Using the full AIDS dataset with three relation types, Figure 6 shows FGWAlign$_{\text{rel}}$ significantly outperforms competitors, including the state-of-the-art TaGSim. FGWAlign$_{\text{rel}}$ reduces RMSE from 3.135 to 1.207 (61.5% decrease) compared to TaGSim, while improving p@20 from 0.322 to 0.700. Compared to FGWAlign$_{\text{full}}$ (which treats all edge types as equivalent), FGWAlign$_{\text{rel}}$ achieves substantial improvements (RMSE: 1.645 $\longrightarrow$ 1.207, p@20: 0.494 $\longrightarrow$ 0.700) with only 60% more computation time, showing its effectiveness in capturing edge type nuances.

5.5 Downstream Applications of FGWAlign

Labeled Graph Alignment. We extend our method beyond GED computation to address graph alignment problems with node and edge labels. Following the same experimental protocol as the current state-of-the-art GABoost approach [40], we evaluate FGWAlign on three graph alignment datasets comprising a total of 12 graph pairs with 1,000-10,000 nodes. For brevity, we direct readers to the

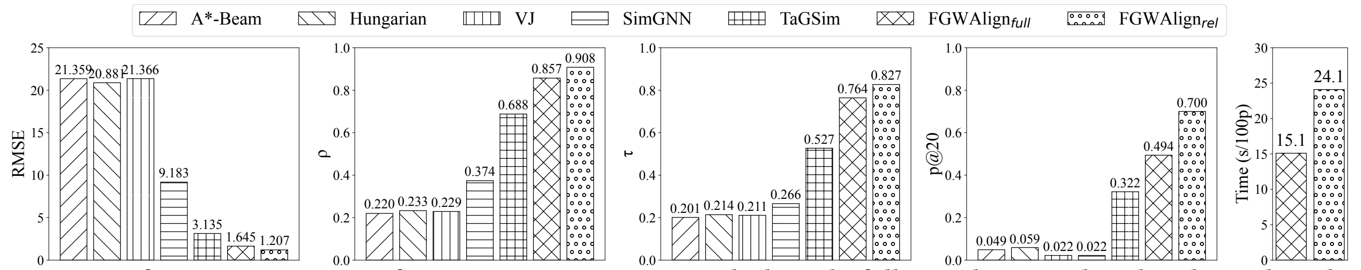


Figure 6: Performance comparison of various GED computation methods on the full AIDS dataset with multi-relational graphs. The rightmost plot shows the computation time for FGWAlign$_{\text{full}}$ and FGWAlign$_{\text{rel}}$.

Dataset	Douban [83]				Movies [40]				Megadiff changes [40]			
Metric	ACC	MAP	EC	ICS	ACC	MAP	EC	ICS	ACC	MAP	EC	ICS
FINAL [83]	42.4	55.0	6.8	36.5	66.1	2.9	49.0	51.4	0.2	0.9	2.8	2.8
REGAL [25]	5.6	19.5	0.8	4.4	10.8	21.4	2.4	2.5	50.7	60.5	68.5	69.1
WAlign [20]	66.5	59.4	15.6	59.3	63.9	63.4	43.6	45.7	48.6	20.4	55.0	55.6
NAME [26]	35.7	61.6	7.7	41.8	92.5	91.2	72.9	76.3	49.3	51.0	71.6	72.2
GTCAlign [70]	60.8	68.4	14.1	76.2	82.7	82.9	62.8	65.8	57.9	63.3	81.4	82.1
SLOTAlign [64]	51.5	61.3	13.0	62.3	90.5	91.6	67.4	70.5	50.3	33.8	72.1	72.9
+GABoost [40]	79.8	97.3	17.3	93.3	**96.5**	94.1	71.7	75.1	**61.0**	**76.7**	88.1	89.0
FGWAlign	75.9	97.8	18.3	98.9	96.4	**94.3**	**73.0**	**76.5**	60.3	76.2	95.9	94.0
+GABoost	**84.2**	**98.6**	**18.4**	**99.4**	**96.5**	94.1	71.7	75.1	60.4	75.8	**96.3**	**96.7**

Table 5: Performance comparison of different graph alignment methods in terms of Accuracy (ACC), Mean Average Precision (MAP), Edge Correctness (EC), and Induced Conserved Structure (ICS). The reported values are scaled to $[0, 100]$.

Dataset	PROTEINS	ENZYMES	AIDS	BZR	DD	NCI1
PK-OCSVM [46, 50]	50.49±4.92	53.67±2.66	50.79±4.30	46.85±5.31	48.30±3.98	49.90±1.18
PK-iF [37, 50]	60.70±2.55	51.30±2.01	51.84±2.87	55.32±6.18	71.32±2.41	50.58±1.38
WL-OCSVM [46, 62]	51.35±4.35	55.24±2.66	50.12±3.43	50.56±5.87	47.99±4.09	50.63±1.22
WL-iF [37, 62]	61.36±2.54	51.60±3.81	61.13±0.71	52.46±3.30	70.31±1.09	50.74±1.70
OCGIN [84]	70.89±2.44	58.75±5.98	78.16±3.05	65.91±1.47	72.27±1.83	71.98±1.21
GLocalKD [44]	77.30±5.15	61.39±8.81	93.27±4.19	69.42±7.78	80.12±5.24	68.48±2.39
OCGTL [57]	76.51±1.55	62.06±3.36	99.40±0.57	63.94±8.89	79.48±2.02	73.44±0.97
SIGNET [41]	75.22±3.91	62.96±4.22	97.27±1.17	**81.44**±9.23	72.72±3.91	**74.89**±2.07
FGWAlign	**77.97**±1.09	**71.11**±6.07	**99.81**±0.07	70.01±2.72	**82.09**±2.47	71.21±1.42

Table 6: Graph-level anomaly detection performance in terms of *ROC-AUC* (in percent, mean ± std). The best results are highlighted with bold.

appendix in our GitHub repository[3] or the GABoost paper [40] for comprehensive implementation details.

Results in Table 5 demonstrate that FGWAlign surpasses all compared methods in structural metrics across three datasets, particularly in edge correctness and induced conserved structure. When enhanced with GABoost's post-processing step, FGWAlign+GABoost achieves superior performance across all metrics on the Douban dataset. These results highlight FGWAlign's versatility in handling both GED computation and node alignment effectively without requiring significant modifications.

Graph-level Anomaly Detection. To demonstrate FGWAlign's applicability beyond graph matching, we apply it to graph-level anomaly detection—identifying test graphs that do not belong to any class in the training set. Our approach computes GED between test and training graphs, assigning higher anomaly scores to test graphs whose closest match in the training set has a larger GED. To reduce computational overhead, we employ the pivot algorithm [10] to select 20 representative candidate graphs among all training graphs for each test graph before applying FGWAlign. The implementation details can be found in our appendix.

We adhere to the same experimental protocol as the state-of-the-art learning-based method SIGNET [41], evaluating performance across six TU datasets [48] via 5-fold cross-validation. For comparison, we include four classic methods (PK-OCSVM, PK-iF, WL-OCSVM, WL-iF) and four GNN-based approaches (OCGIN, GLocalKD, OCGTL, SIGNET) alongside FGWAlign.Table 6 presents the Receiver Operating Characteristic Area Under the Curve (ROC-AUC) metrics, which are calculated based on anomaly scores and corresponding labels [84]. FGWAlign achieves state-of-the-art performance on four of the six datasets (PROTEINS, ENZYMES, AIDS, and DD), with particularly notable improvements on ENZYMES (8.15% higher than the next best method) and competitive performance on BZR and NCI1. These results validate that FGWAlign effectively generalizes to anomaly detection tasks. Despite its enhanced interpretability, it outperforms specialized deep learning methods, underscoring its practical utility.

6 CONCLUSION

This paper approaches the four-decade-old NP-complete problem of GED computation from a novel perspective. We propose FGWAlign, a framework that reformulates GED computation within the optimal transport paradigm by leveraging the Fused Gromov-Wasserstein distance. Through three key enhancements—including a diverse projection strategy, a random exploration scheme, and a multi-relational extension—FGWAlign achieves superior computation quality, efficiency, and scalability compared to existing methods. We further validate FGWAlign's effectiveness in two downstream applications: graph alignment and graph-level anomaly detection. These results underscore how FGWAlign, by enabling more accurate GED-based similarity estimation, unlocks potential for broader applications in graph data menagement.

ACKNOWLEDGMENTS

The research work described in this paper was supported by Guangdong S&T Program C019 and Hong Kong Research Grants Council (grant# 16202722, T22-607/24-N, T43-513/23N-1). It was partially conducted in JC STEM Lab of Data Science Foundations funded by The Hong Kong Jockey Club Charities Trust.

[3]https://github.com/squareRoot3/FGWAlign/blob/main/appendix.pdf

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 *4th International Conference on Pattern Recognition Applications and Methods 2015*.

[2] HA Almohamad and Salih O Duffuaa. 1993. A linear programming approach for the weighted graph matching problem. *IEEE Transactions on pattern analysis and machine intelligence* 15, 5 (1993), 522–525.

[3] Jiyang Bai and Peixiang Zhao. 2022. TaGSim: type-aware graph similarity learning and computation. *Proceedings of the VLDB Endowment* 15, 2 (2022).

[4] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. 2019. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the twelfth ACM international conference on web search and data mining*. 384–392.

[5] David B Blumenthal, Nicolas Boria, Johann Gamper, Sébastien Bougleux, and Luc Brun. 2020. Comparing heuristics for graph edit distance computation. *The VLDB journal* 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 *International Workshop on Graph-Based Representations in Pattern Recognition*. Springer, 211–221.

[7] Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. 2011. Displacement interpolation using Lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia conference*. 1–12.

[8] Sebastien Bougleux, Luc Brun, Vincenzo Carletti, Pasquale Foggia, Benoit Gaüzère, and Mario Vento. 2017. Graph edit distance as a quadratic assignment problem. *Pattern Recognition Letters* 87 (2017), 38–46.

[9] Horst Bunke and Kim Shearer. 1998. A graph distance metric based on the maximal common subgraph. *Pattern recognition letters* 19, 3-4 (1998), 255–259.

[10] B. Bustos, G. Navarro, and E. Chavez. 2001. Pivot selection techniques for proximity searching in metric spaces. In *SCCC 2001. 21st International Conference of the Chilean Computer Science Society*. 33–40. https://doi.org/10.1109/SCCC.2001.972629

[11] Lijun Chang, Xing Feng, Xuemin Lin, Lu Qin, Wenjie Zhang, and Dian Ouyang. 2020. Speeding up GED verification for graph similarity search. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 793–804.

[12] Liqun Chen, Zhe Gan, Yu Cheng, Linjie Li, Lawrence Carin, and Jingjing Liu. 2020. Graph optimal transport for cross-domain alignment. In *International Conference on Machine Learning*. PMLR, 1542–1553.

[13] Qihao Cheng, Da Yan, Tianhao Wu, Zhongyi Huang, and Qin Zhang. 2025. Computing Approximate Graph Edit Distance via Optimal Transport. *Proceedings of the ACM on Management of Data* 3, 1 (2025), 1–26.

[14] Minsu Cho, Jungmin Lee, and Kyoung Mu Lee. 2010. Reweighted random walks for graph matching. In *European conference on Computer vision*. Springer, 492–505.

[15] Marco Cuturi. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf

[16] Qijie Ding, Daokun Zhang, and Jie Yin. 2022. Conflict-aware pseudo labeling via optimal transport for entity alignment. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 915–920.

[17] Stefan Fankhauser, Kaspar Riesen, and Horst Bunke. 2011. Speeding up graph edit distance computation through fast bipartite matching. In *Graph-Based Representations in Pattern Recognition: 8th IAPR-TC-15 International Workshop, GbRPR 2011, Münster, Germany, May 18-20, 2011. Proceedings 8*. Springer, 102–111.

[18] Andreas Fischer, Ching Y Suen, Volkmar Frinken, Kaspar Riesen, and Horst Bunke. 2015. Approximation of graph edit distance based on Hausdorff matching. *Pattern Recognition* 48, 2 (2015), 331–343.

[19] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, et al. 2021. Pot: Python optimal transport. *Journal of Machine Learning Research* 22, 78 (2021), 1–8.

[20] Ji Gao, Xiao Huang, and Jundong Li. 2021. Unsupervised graph alignment with wasserstein distance discriminator. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 426–435.

[21] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. 2010. A survey of graph edit distance. *Pattern Analysis and applications* 13 (2010), 113–129.

[22] Carlos Garcia-Hernandez, Alberto Fernandez, and Francesc Serratosa. 2019. Ligand-based virtual screening using graph edit distance as molecular similarity measure. *Journal of chemical information and modeling* 59, 4 (2019), 1410–1421.

[23] Johannes Gasteiger, Marten Lienen, and Stephan Günnemann. 2021. Scalable optimal transport in high dimensions for graph distances, embedding alignment, and more. In *International Conference on Machine Learning*. PMLR, 5616–5627.

[24] Karam Gouda and Mosab Hassaan. 2016. CSI_GED: An efficient approach for graph edit similarity computation. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 265–276.

[25] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. Regal: Representation learning-based graph alignment. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 117–126.

[26] Thanh Trung Huynh, Chi Thang Duong, Thanh Tam Nguyen, Vinh Tong Van, Abdul Sattar, Hongzhi Yin, and Quoc Viet Hung Nguyen. 2021. Network alignment with holistic embeddings. *IEEE Transactions on Knowledge and Data Engineering* 35, 2 (2021), 1881–1894.

[27] Rashid Ibragimov, Maximilian Malek, Jiong Guo, and Jan Baumbach. 2013. Gedevo: an evolutionary graph edit distance algorithm for biological network alignment. In *German conference on bioinformatics 2013*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[28] Roy Jonker and Ton Volgenant. 1988. A shortest augmenting path algorithm for dense and sparse linear assignment problems. In *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR*. Springer, 622–622.

[29] Derek Justice and Alfred Hero. 2006. A binary linear programming formulation of the graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28, 8 (2006), 1200–1214.

[30] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, et al. 2023. PubChem 2023 update. *Nucleic acids research* 51, D1 (2023), D1373–D1380.

[31] Nils M Kriege, Pierre-Louis Giscard, Franka Bause, and Richard C Wilson. 2019. Computing optimal assignments in linear time for approximate graph matching. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 349–358.

[32] Marius Leordeanu and Martial Hebert. 2005. A spectral technique for correspondence problems using pairwise constraints. In *International Conference on Computer Vision*. IEEE, 1482–1489.

[33] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. 2009. An integer projected fixed point method for graph matching and map inference. *Advances in neural information processing systems* 22 (2009).

[34] Jiajin Li, Jianheng Tang, Lemin Kong, Huikang Liu, Jia Li, Anthony Man-Cho So, and Jose Blanchet. 2022. Fast and Provably Convergent Algorithms for Gromov-Wasserstein in Graph Data. arXiv:2205.08115 [cs.LG] https://arxiv.org/abs/2205.08115

[35] Jiajin Li, Jianheng Tang, Lemin Kong, Huikang Liu, Jia Li, Anthony Man-Cho So, and Jose Blanchet. 2023. A Convergent Single-Loop Algorithm for Relaxation of Gromov-Wasserstein in Graph Data. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=0jxPyVWmiiF

[36] Chih-Long Lin. 1994. Hardness of approximating graph transformation problem. In *International Symposium on Algorithms and Computation*. Springer, 74–82.

[37] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 413–422.

[38] Junbin Liu, Ya Liu, Wing-Kin Ma, Mingjie Shao, and Anthony Man-Cho So. 2024. Extreme Point Pursuit – Part I: A Framework for Constant Modulus Optimization. arXiv:2403.06506 [eess.SP]

[39] 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*. 1503–1512.

[40] Wei Liu, Wei Zhang, Haiyan Zhao, and Zhi Jin. 2024. GABoost: Graph Alignment Boosting via Local Optimum Escape. *Proc. ACM Manag. Data* 2, 4, Article 199 (Sept. 2024), 26 pages. https://doi.org/10.1145/3677135

[41] Yixin Liu, Kaize Ding, Qinghua Lu, Fuyi Li, Leo Yu Zhang, and Shirui Pan. 2023. Towards self-interpretable graph-level anomaly detection. In *Advances in Neural Information Processing Systems*, Vol. 36.

[42] Zhi-Yong Liu and Hong Qiao. 2014. GNCCP—Graduated NonConvexity and Concavity Procedure. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 6 (2014), 1258–1267. https://doi.org/10.1109/TPAMI.2013.223

[43] Shengxuan Luo and Sheng Yu. 2022. An Accurate Unsupervised Method for Joint Entity Alignment and Dangling Entity Detection. In *Findings of the Association for Computational Linguistics: ACL 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 2330–2339. https://doi.org/10.18653/v1/2022.findings-acl.183

[44] Rongrong Ma, Guansong Pang, Ling Chen, and Anton van den Hengel. 2022. Deep graph-level anomaly detection by glocal knowledge distillation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 704–714.

[45] Paul Maergner, Vinaychandran Pondenkandath, Michele Alberti, Marcus Liwicki, Kaspar Riesen, Rolf Ingold, and Andreas Fischer. 2019. Combining graph edit distance and triplet networks for offline signature verification. *Pattern Recognition Letters* 125 (2019), 527–533.

[46] Larry M Manevitz and Malik Yousef. 2001. One-class SVMs for document classification. *Journal of machine Learning research* 2, Dec (2001), 139–154.

[47] Facundo Mémoli. 2009. Spectral Gromov-Wasserstein distances for shape matching. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE, 256–263.

[48] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML Workshop*.

[49] Michel Neuhaus, Kaspar Riesen, and Horst Bunke. 2006. Fast suboptimal algorithms for the computation of graph edit distance. In *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, SSPR 2006 and SPR 2006, Hong Kong, China, August 17-19, 2006. Proceedings*. Springer, 163–172.
[50] Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. 2016. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning* 102 (2016), 209–245.
[51] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. 2017. Matching node embeddings for graph similarity. In *Proceedings of the AAAI conference on Artificial Intelligence*, Vol. 31.
[52] Shichao Pei, Lu Yu, and Xiangliang Zhang. 2019. Improving cross-lingual entity alignment via optimal transport. In *IJCAI*.
[53] Yun Peng, Byron Choi, and Jianliang Xu. 2021. Graph Edit Distance Learning via Modeling Optimum Matchings with Constraints.. In *IJCAI*. 1534–1540.
[54] Gabriel Peyré, Marco Cuturi, and Justin Solomon. 2016. Gromov-Wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*. PMLR, 2664–2672.
[55] Chengzhi Piao, Tingyang Xu, Xiangguo Sun, Yu Rong, Kangfei Zhao, and Hong Cheng. 2023. Computing Graph Edit Distance via Neural Graph Matching. *Proceedings of the VLDB Endowment* 16, 8 (2023), 1817–1829.
[56] Zongyue Qin, Yunsheng Bai, and Yizhou Sun. 2020. GHashing: Semantic graph hashing for approximate similarity search in graph databases. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2062–2072.
[57] Chen Qiu, Marius Kloft, Stephan Mandt, and Maja Rudolph. 2022. Raising the Bar in Graph-level Anomaly Detection. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*. 2196–2203.
[58] Kaspar Riesen and Horst Bunke. 2009. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision computing* 27, 7 (2009), 950–959.
[59] Kaspar Riesen, Stefan Fankhauser, and Horst Bunke. 2007. Speeding Up Graph Edit Distance Computation with a Bipartite Heuristic.. In *Mining and Learning with Graphs*. 21–24.
[60] Kaspar Riesen, Michel Neuhaus, and Horst Bunke. 2007. Bipartite graph matching for computing the edit distance of graphs. In *Graph-Based Representations in Pattern Recognition: 6th IAPR-TC-15 International Workshop, GbRPR 2007, Alicante, Spain, June 11-13, 2007. Proceedings 6*. Springer, 1–12.
[61] Alberto Sanfeliu and King-Sun Fu. 1983. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics* 3 (1983), 353–362.
[62] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12, 9 (2011).
[63] Justin Solomon, Gabriel Peyré, Vladimir G Kim, and Suvrit Sra. 2016. Entropic metric alignment for correspondence problems. *ACM Transactions on Graphics (ToG)* 35, 4 (2016), 1–13.
[64] Jianheng Tang, Weiqi Zhang, Jiajin Li, Kangfei Zhao, Fugee Tsung, and Jia Li. 2023. Robust Attributed Graph Alignment via Joint Structure Learning and Optimal Transport. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. 1638–1651. https://doi.org/10.1109/ICDE55515.2023.00129
[65] Jianheng Tang, Kangfei Zhao, and Jia Li. [n.d.]. A Fused Gromov-Wasserstein Framework for Unsupervised Knowledge Graph Entity Alignment. In *Findings of the Association for Computational Linguistics: ACL 2023*. 3320–3334. https://doi.org/10.18653/v1/2023.findings-acl.205
[66] Vayer Titouan, Nicolas Courty, Romain Tavenard, and Rémi Flamary. 2019. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*. PMLR, 6275–6284.
[67] Michel Van Kempen, Stephanie S Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron LM Gilchrist, Johannes Söding, and Martin Steinegger. 2023. Fast and accurate protein structure search with Foldseek. *Nature Biotechnology* (2023), 1–4.
[68] Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, et al. 2022. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic acids research* 50, D1 (2022), D439–D444.
[69] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. *The Journal of Machine Learning Research* 11 (2010), 1201–1242.
[70] Chenxu Wang, Peijing Jiang, Xiangliang Zhang, Pinghui Wang, Tao Qin, and Xiaohong Guan. 2023. GTCAlign: Global Topology Consistency-based Graph Alignment. *IEEE Transactions on Knowledge and Data Engineering* (2023).
[71] Runzhong Wang, Ziao Guo, Wenzheng Pan, Jiale Ma, Yikai Zhang, Nan Yang, Qi Liu, Longxuan Wei, Hanxue Zhang, Chang Liu, Zetian Jiang, Xiaokang Yang, and Junchi Yan. 2024. Pygmtools: A Python Graph Matching Toolkit. *Journal of Machine Learning Research* 25, 33 (2024), 1–7. https://jmlr.org/papers/v25/23-0572.html
[72] Runzhong Wang, Tianqi Zhang, Tianshu Yu, Junchi Yan, and Xiaokang Yang. 2021. Combinatorial learning of graph edit distance via dynamic embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5241–5250.
[73] Xiaoli Wang, Xiaofeng Ding, Anthony KH Tung, Shanshan Ying, and Hai Jin. 2012. An efficient graph indexing method. In *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 210–221.
[74] Richard C Wilson and Ping Zhu. 2008. A study of graph spectra for comparing graphs and trees. *Pattern Recognition* 41, 9 (2008), 2833–2841.
[75] Kexuan Xin, Zequn Sun, Wen Hua, Wei Hu, Jianfeng Qu, and Xiaofang Zhou. 2022. Large-scale Entity Alignment via Knowledge Graph Merging, Partitioning and Embedding. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2240–2249.
[76] Hongteng Xu, Dixin Luo, Hongyuan Zha, and Lawrence Carin Duke. 2019. Gromov-wasserstein learning for graph matching and node embedding. In *International conference on machine learning*. PMLR, 6932–6941.
[77] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1365–1374.
[78] Lei Yang and Lei Zou. 2021. Noah: Neural-optimized A Search Algorithm for Graph Edit Distance Computation. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 576–587.
[79] Ye Yuan, Guoren Wang, Lei Chen, and Haixun Wang. 2012. Efficient subgraph similarity search on large probabilistic graph databases. *Proc. VLDB Endow.* 5, 9 (May 2012), 800–811. https://doi.org/10.14778/2311906.2311908
[80] Li Yujian and Liu Bo. 2007. A normalized Levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence* 29, 6 (2007), 1091–1095.
[81] Weixin Zeng, Xiang Zhao, Xinyi Li, Jiuyang Tang, and Wei Wang. 2022. On entity alignment at scale. *The VLDB Journal* (2022), 1–25.
[82] Zhiping Zeng, Anthony KH Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. 2009. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment* 2, 1 (2009), 25–36.
[83] Si Zhang and Hanghang Tong. 2016. Final: Fast attributed network alignment. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1345–1354.
[84] Lingxiao Zhao and Leman Akoglu. 2021. On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights. *Big Data* (2021).
[85] Xiang Zhao, Chuan Xiao, Xuemin Lin, Qing Liu, and Wenjie Zhang. 2013. A partition-based approach to structure similarity search. *Proceedings of the VLDB Endowment* 7, 3 (2013), 169–180.
[86] Xiang Zhao, Chuan Xiao, Xuemin Lin, and Wei Wang. 2012. Efficient graph similarity joins with edit distance constraints. In *2012 IEEE 28th international conference on data engineering*. IEEE, 834–845.
[87] Xiang Zhao, Chuan Xiao, Xuemin Lin, Wenjie Zhang, and Yang Wang. 2018. Efficient structure similarity searches: a partition-based approach. *The VLDB Journal* 27, 1 (2018), 53–78.
[88] Weiguo Zheng, Lei Zou, Xiang Lian, Dong Wang, and Dongyan Zhao. 2014. Efficient graph similarity search over large graph databases. *IEEE Transactions on Knowledge and Data Engineering* 27, 4 (2014), 964–978.