
# A Semantics-aware Approach for Graph Edit Distance Estimation over Knowledge Graphs

Yingli Zhou
The Chinese University of Hong Kong, Shenzhen
yinglizhou@link.cuhk.edu.cn

Huizhong Wang
The Chinese University of Hong Kong, Shenzhen
huizhongwang@link.cuhk.edu.cn

Chenhao Ma
The Chinese University of Hong Kong, Shenzhen
machenhao@cuhk.edu.cn

Yixiang Fang*
The Chinese University of Hong Kong, Shenzhen
fangyixiang@cuhk.edu.cn

## ABSTRACT

Graph Edit Distance (GED) is a key metric for measuring the similarity between two Knowledge Graphs (KGs), defined as the minimum number of atomic operations required to transform one KG into another. It has broad applications in fields such as pattern recognition, biological analysis, and graph databases. The state-of-the-art approaches adopt Graph Neural Networks (GNNs) to predict GED, but they are limited to simple graphs and cannot be directly applied to the KGs, as they fail to capture the rich semantics and complex relationships present in KGs. To design a KG-native solution, in this paper, we propose a semantics-aware GNN model, SEABED, to capture local semantic dependencies and global semantic consistency between two KGs. Extensive experiments on four real-world KGs demonstrate that our proposed algorithm outperforms the state-of-the-art methods on all datasets. In particular, the mean absolute error is reduced by up to 66.7%, while the accuracy is improved by up to 70.5%, without increasing the computation time.

**PVLDB Reference Format:**
Yingli Zhou, Huizhong Wang, Chenhao Ma, and Yixiang Fang. A Semantics-aware Approach for Graph Edit Distance Estimation over Knowledge Graphs. PVLDB, 19(6): 1226 - 1239, 2026.
doi:10.14778/3797919.3797930

**PVLDB Artifact Availability:**
The source code, data, and/or other artifacts have been made available at https://github.com/HuiZ-W/SEABED

## 1 INTRODUCTION

Knowledge Graph (KG) stands out as one of the famous graph data types [28], via semi-structured data models like the Resource Description Framework (RDF) [14]. In KGs, labeled nodes typically correspond to entities, and directed, labeled edges between nodes denote the relationships between the entities. Figure 1 gives a toy example. Nowadays, KGs are fueling diverse applications like question-answering, recommendation systems, and semantic databases [53].

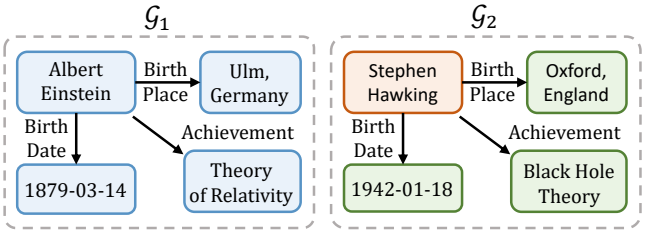



**Figure 1: An example pair of KGs with GED=4.**

Graph Edit Distance (GED) is a fundamental definition in graph analysis, which has been extensively studied in the literature [4, 5, 10, 17, 21, 29, 36, 41, 54, 56, 57, 75, 86]. Similar to the edit distance on strings, the GED on a pair of graphs is defined as the minimum number of atomic operations that can convert one graph to the other. These atomic operations, which consist of node and edge insertions, deletions, or relabelings, form a sequence known as the *graph edit path*. The length of this path is the GED. In this paper, we focus on accurately estimating GED between a pair of KGs. Figure 1 depicts two KGs, $\mathcal{G}_1$ and $\mathcal{G}_2$. We can transform $\mathcal{G}_1$ into $\mathcal{G}_2$ with four edit operations—specifically, by changing four entities in the sub-KG related to "Albert Einstein" to those corresponding to "Stephen Hawking". Therefore, the GED between $\mathcal{G}_1$ and $\mathcal{G}_2$ is 4. Due to the prevalence of KGs, the GED, as a commonly used metric to estimate the "similarities" between two KGs, has found various applications in a wide range of fields, including pattern recognition [9, 18], biological analysis [19, 20, 44, 54], and graph databases [41, 62]. For example, in the drug design process, GED helps in comparing molecular structures represented as KGs to evaluate their similarity. Specifically, GED can thus be employed to perform virtual screening of molecules, leading to the identification of structurally similar compounds that may exhibit similar pharmacological properties [19, 20]. Moreover, in modern graph database systems, graph search queries [4, 5, 16, 41] often rely on GED computations over KGs.

• **Prior works.** While computing the GED over KGs is highly valuable, we found that all existing works are designed for general graphs rather than specifically for KGs. It is well known that computing the GED is an NP-hard problem [10] in graph theory, making exact algorithms typically intractable when the graph size exceeds 16 nodes [49]. Hence, recent studies have paid more attention to approximate GED computation. The representative approximation algorithms mainly rely on machine learning techniques [4, 5, 54, 83]. Specifically, Bai et al. [5] first propose an end-to-end GNN-based

*Yixiang Fang is the corresponding author.


Proceedings of the VLDB Endowment, Vol. 19, No. 6 ISSN 2150-8097.
doi:10.14778/3797919.3797930

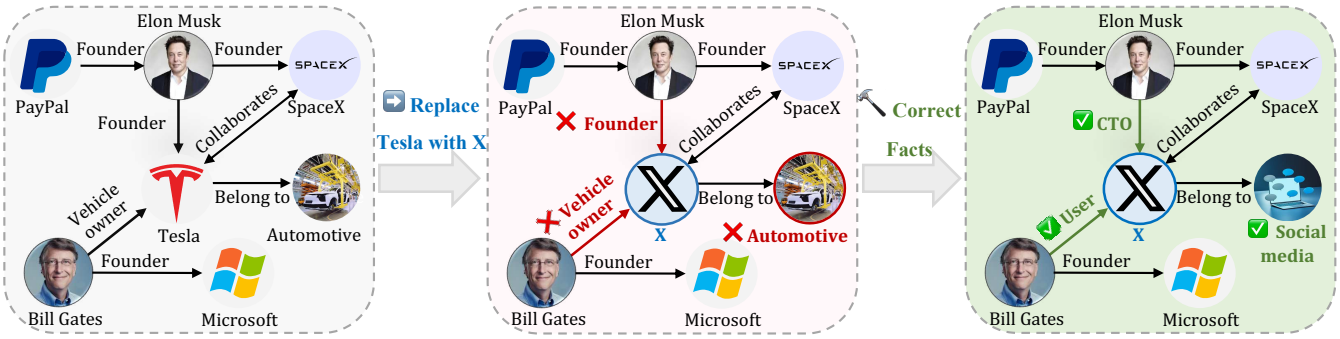



**Figure 2: An example illustrating the challenge of computing GED between a pair of KGs.**

method SimGNN, which first employs GNN to capture the graph-level information of each graph, combines them into a single vector to measure the difference between inputs, and then feeds it into a Multi-Layer Perceptron (MLP) model to estimate the GED of two graphs. To improve the accuracy, GEDGNN [54] integrates minimum weight matching [29] into its training pipeline and applies a post-processing refinement step. GEDIOT [12], the current state-of-the-art, extends GEDGNN by employing optimal transport-based alignment, leading to more accurate estimation.

• **Key challenges of designing KG-native method.** While existing methods have achieved promising results on general graphs, directly applying them to KGs may degrade performance, due to the rich semantics and complex relationships present in KGs. To design a KG-native method, two key challenges need to be solved:
**Challenge 1: Semantic entanglement of local subgraphs.** In general graphs, if two graphs differ by a single node label, their GED is one. However, this simple correspondence breaks down in KGs, where maintaining semantic consistency and preserving the logical relationships among facts introduce additional constraints. For example, as shown in Figure 2, replacing *Tesla* with *X* in a KG about "Elon Musk" disrupts semantic coherence and introduces factual errors. Specifically, the Tesla-related facts become inconsistent: triples such as $\langle ElonMusk, founder, X\rangle$ and $\langle X, belong\ to, Automotive\rangle$ are factually incorrect and do not reflect reality. In other words, a single edit in the KG may invalidate several related facts, making GED estimation in KGs far more challenging than in general graphs.
**Challenge 2: Global semantic inconsistency.** KGs possess global structural semantics, such as type hierarchies (e.g., "Car Company" as a subtype of "Company") and role constraints (e.g., "isOwnerOf" should link a "Person" to a "Company"), which existing methods cannot align high-level semantic distributions between KGs.

• **Our technical contributions.** To address the above challenges, we propose SEABED, a novel semantics-aware GNN-based GED estimation method. Specifically, to tackle **Challenge 1**, we begin by partitioning each KG into overlapping subgraphs (sub-KGs) using a semantics-driven partitioning algorithm, ensuring that semantically entangled triples are grouped within the same subgraph. Each sub-KG is first encoded by a GNN to produce a high-level representation. By treating each sub-KG as a semantic token, we then apply self-attention across all sub-KGs from both KGs, enabling the model to capture complex semantic interactions and learn adaptive alignments between sub-KG pairs. This mechanism emphasizes the most semantically relevant subgraph correspondences while down-weighting irrelevant or noisy matches. By doing so, our model captures subgraph semantics and dynamically models inter-dependencies among sub-KGs for GED estimation.

To address **Challenge 2**, SEABED first generates atom-level embeddings for entities and predicates using RDF2Vec [14]. These embeddings are then processed by a GNN with a KG-native message-passing strategy that captures both entity correlations and edge directionality. The GNN produces comprehensive graph-level representations for each KG. To assess global similarity between KGs, we compute the Earth Mover's Distance (EMD) [60] between their semantic embeddings. EMD provides a principled approach to align and compare overall semantic distributions, ensuring that global semantic consistency is preserved in GED computation. Such a two-stage design allows our model to first extract fine-grained semantic and structural signals at both the local (sub-KG) and global (KG) levels, and then holistically capture their complex inter-dependencies for robust GED estimation. A notable feature of SEABED is its semantics-awareness: it captures both local and global semantic information, as well as structural differences, making it truly KG-native.

Extensive experimental evaluations on four real-world KGs show that SEABED achieves higher estimation accuracy than the state-of-the-art algorithm on all datasets. Particularly, the Mean Absolute Error is reduced by as much as 66.7%, while accuracy is improved by up to 70.5% without increasing the computation time. In summary, our main contributions are as follows.

- To the best of our knowledge, we are the first to propose a KG-native learning-based model to estimate the GED over a pair of KGs.
- We design a semantics-aware GNN model to capture local semantic dependencies and global semantic consistency between two KGs.
- We conduct experiments on four real-world KGs to demonstrate the effectiveness of our model.

**Outline.** We introduce the preliminaries in Section 2. Section 3 analyzes the limitations of SOTA algorithms. We introduce our SEABED model in Section 4. The experimental results are reported in Section 5. We review the related work in Section 6 and conclude in Section 7.

## 2 PRELIMINARIES

### 2.1 Problem definition

A knowledge graph (KG) is defined as a directed graph, where nodes and edges are associated with labels [26]. Several data models are used to represent KGs. In this paper, we follow the classical and widely-used definition of the Resource Description Framework (RDF) [14], where every statement is modeled as a triple.

*Definition 2.1 (Knowledge Graph [26]).* A Knowledge Graph (KG) is a directed labeled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, and $\mathcal{E}$ is the set of edges. KG is represented as a set of triples, where each triple is of the form $(h, p, t)$, with $h$ (the head) and $t$ (the tail) denoting entities, and $p$ representing the predicate (relation) between them. The entities $h$ and $t$ can carry labels to denote their classes, e.g., human, organization, etc. The predicate $p$ can be treated as the labeled edge. The elements of a triple are referred to as atoms.

Denote by $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$ the number of nodes and edges in $\mathcal{G}$, respectively. For each node $x_i \in \mathcal{V}$, let $h_i \in \mathbb{R}^d$ denote its $d$-dimensional high-level embedding, and let $N(x_i, \mathcal{G})$ represent its set of neighbors in $\mathcal{G}$. For each directed edge $e_{i,j} \in \mathcal{E}$ from node $x_i$ to node $x_j$, let $h_{i,j} \in \mathbb{R}^d$ denote its $d$-dimensional embedding.

*Definition 2.2 (Graph Edit Distance [10]).* Given a pair of KGs $(\mathcal{G}_1, \mathcal{G}_2)$, Graph Edit Distance (GED) is the minimum number of operations required to transform $\mathcal{G}_1$ into $\mathcal{G}_2$, denoted as $d(\mathcal{G}_1, \mathcal{G}_2)$. Specifically, there are three types of primitive operations: (1) adding or removing an edge; (2) adding or removing an isolated node, and (3) changing the label of an edge or a node.

KGs are typically very large [76], making the alignment of two entire KGs infeasible and often lacking practical significance. Instead, the focus is on aligning meaningful small KGs, such as family relationships in a social network [48] or collaborations in a specific research domain [87]. In this paper, we primarily focus on measuring the similarity between such small KGs.

> **Problem statement.** In this work, we aim to train a machine learning-based model to estimate the GED between two KGs, $\mathcal{G}_1$ and $\mathcal{G}_2$, denoted as $\widehat{d}(\mathcal{G}_1, \mathcal{G}_2)$.

Example 1. *Figure 1 gives an example of two KGs, $\mathcal{G}_1$ and $\mathcal{G}_2$. In this case, we can transform $\mathcal{G}_1$ into $\mathcal{G}_2$ by changing four entities, hence the GED over $\mathcal{G}_1$ and $\mathcal{G}_2$ is $d(\mathcal{G}_1, \mathcal{G}_2) = 4$.*

### 2.2 Knowledge graph embedding

Knowledge graph embeddings (KGE) are low-dimensional vectors of the atoms (entities and predicates) in a KG. These vectors encode latent semantic information about the atoms that are not explicitly available in the KG, which are widely used in downstream tasks [59], including link prediction, entity classification, clustering, etc.

Due to differences in components and learning processes, various KGE models can capture different latent aspects of a KG, making certain KGE models more suitable for specific tasks. For example, some KGE models (e.g., translation-based models [59]) learn from individual triples in the KG by embedding entities and relationships into the same vector space, allowing the model to infer possible links based on geometric relationships between vectors. Such models are well-suited for link prediction tasks. Other models (e.g., RDF2Vec [64] or Ridle [78] ) can learn from larger structures like subgraphs, making them more appropriate for tasks involving the interpretation of semantic relatedness and similarity of atoms.

In this paper, we employ the latter model, i.e., RDF2Vec to obtain the KGE, since it is better to capture the semantic and structural correlations in the KG. Specifically, RDF2Vec is a Skip-gram model [47] based on random walks over atoms in the KG. By starting from any node and traversing the KG along its edges, it generates random walks on the KG. Given a random walk of atoms $w_1, w_2, \cdots, w_L$ in a KG of length $L$, the objective of RDF2Vec is to maximize the following log probability (i.e. minimize its negative), which can be seen as the loss function in KGE:

$$\frac{1}{L}\sum_{t=1}^{L}\sum_{-c\le j\le c, j\neq 0}\log p(w_{t+j}|w_t), \tag{1}$$

where $c$ denotes the context window size, which controls how many neighboring nodes in the random walk sequence are considered.

Here, a notable feature of the embeddings generated by RDF2Vec is that: it encodes the atom's relatedness and correlates with the joint probability of observing it. A single entity participates in many walks, and rich contextual information from the neighboring nodes and edges is encoded. Hence, RDF2Vec embeddings are suitable for atom representation in estimating GED over KGs.

### 2.3 Graph neural network

Graph Neural Network (GNN) [35] is a special form of Neural Network (NN) that is tailored to graph-structured data. The distinct feature of GNNs is that they are invariant with respect to a permutation of the nodes of the input graph. This means that the output of the model is the same even if the atoms in the graph are reordered. Permutation invariance makes GNN parameters and data efficient. The model is parameter efficient as it needs fewer learnable parameters because it is not necessary to learn different permutations of the same graph structure. In the literature, GNNs are widely used for graph-level predictions, where the input is a graph, and the output is a prediction that applies to the entire graph, such as determining whether a molecular graph is toxic or a social network exhibits harmful behavior. That is, predicting the GED between a pair of KGs can be treated as a graph-level prediction task.

GNNs work under the message-passing framework, where in every layer $k$ in the GNN the representations of the nodes are updated based on messages from all connected neighbors. The most generic message-passing function for a node is given by [35]:

$$h_i^{(k)} = \gamma^{(k)}\left(h_i^{(k-1)}, \sum_{x_j\in N(x_i,\mathcal{G})}\phi^{(k)}\left(h_i^{(k-1)} \parallel h_{i,j} \parallel h_j^{(k-1)}\right)\right), \tag{2}$$

where $\gamma$ and $\phi$ are differentiable functions, and for a node $x_i$, its previous representation denoted $h_i^{(k-1)}$ gets non-linearly transformed to a new representation $h_i^{(k)}$ by combining $h_i^{(k-1)}$ with neighboring nodes' representations as well as the edges between them (i.e., the messages).

For graph-level tasks, it is necessary to combine the features of all nodes into a fixed-length vector. To achieve this, max pooling, average pooling, or sum pooling are commonly used functions to aggregate node vectors into a single graph-level vector that

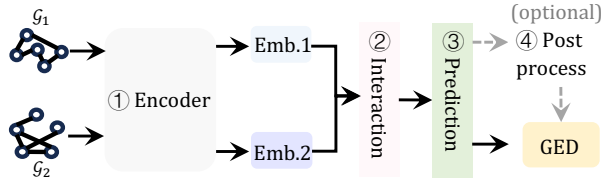


**Figure 3: A general workflow of learning-based approaches.**

represents the whole graph and can be used to predict the desired graph-level quantities [80].

The initial representation $h_i^0$ of a node $x_i$, is typically based on the one-hot or binary encoding to denote the id of the node or more informative features. For example, if the node represents a person in a social graph, one dimension could represent the height of the person, another the age, etc. Here, more meaningful features are superior as they provide some key information to the GNN to solve the desired problem. Note that all existing learning-based GED methods are based on the one-hot encoding to obtain the initial node features. As KGE provides semantic information about the given entity, in our following model design, we use RDF2Vec embeddings as the initial features for each node and predicate.

## 3 ANALYSIS OF SOTA LEARNING-BASED APPROACHES

In this section, we extensively review the state-of-the-art learning-based GED calculation methods.

Recently, due to the powerful representational capacity of neural networks, machine learning-based methods for GED computation have garnered significant attention and achieved considerable success [4, 5, 54, 83]. We surprisingly found that all the existing methods follow the same framework, as shown in Figure 3, which works as follows: (1) an encoder to obtain high-level representations (i.e., embeddings) for the two input KGs; (2) employ an interaction module to capture the structural difference between two KGs, and (3) predict the GED based on the output from the previous step. In addition, some methods incorporate post-processing steps to improve accuracy.

More particularly, Bai et al. [5] first proposed a GNN-based model `SimGNN`, which employs GNN as the encoder, and neural tensor network (NTN) [63] as the interaction module. After training, `SimGNN` can predict the GED of graph pairs in $O(n+m)$ time, where $m$ is the number of edges in the graph. Moreover, Yang et al. [83] proposed using a Graph Isomorphism Network [81] as an encoder to train a model that guides A*-beam search. The method that directly predicts GED based on the trained model is referred to as `GPN`, whereas the approach that utilizes A*-beam search with model guidance is named `Noah`. To improve the accuracy, instead of directly predicting the GED, `TaGSim` [4] focuses on estimating the number of each type of operation.

The `GEDGNN` [54] approach incorporates minimum weight matching into the learning process and designs a post-processing algorithm. Similar to other methods, `GEDGNN` employs GNN as its encoder, but distinguishes itself during the interaction process by using both NTN and cross-matrix calculations on the node embeddings of the two graphs. Besides, it contains a post-processing stage to improve the prediction accuracy by extracting the best matches. The state-of-the-art approach `GEDIOT` [12] shares a similar overall framework with `GEDGNN`, employing both NTN and cross-matrix operations for GED prediction. On the other hand, `GEDIOT` employs the Sinkhorn algorithm [13] for optimal transport in post-processing, further improving node alignment.

> **Motivations.** Existing learning-based methods are designed for simple graphs; directly applying them to KGs may degrade performance due to the rich semantics and complex relationships present in KGs. In this paper, we aim to design a KG-native method to address the two key challenges discussed in Section 1.

## 4 OUR APPROACH

In this section, we introduce our `SEABED` model by first providing an overview of the framework, followed by a detailed discussion of each stage in `SEABED`.

### 4.1 Overview

In this section, we propose a GNN-based method, `SEABED`, to estimate the GED between two KGs efficiently; the overall framework is shown in Figure 4. Specifically, given two KGs $\mathcal{G}_1$ and $\mathcal{G}_2$, our `SEABED` contains three stages: *182 initial embedding extraction*, *183 local semantic alignment*, and *184 global semantic estimator*. The *182 initial embedding extraction* stage, detailed in Section 4.2, involves computing KG embeddings for each knowledge graph using RDF2Vec and enhances these representations by incorporating the local subgraph counts of each node, as capturing this feature is crucial for accurate GED estimation (see Section 4.2).

In the *183 local semantic alignment* phase, we first partition each KG into $k/2$ sub-KGs by intuitively grouping nodes with higher semantic similarity into the same cluster. Thus, we obtain $k$ sub-KGs in total. Afterwards, we treat each sub-KG $\mathcal{S}_i$ $(1 \le i \le k)$ as a "token" and employ a self-attention mechanism to adaptively align semantically related substructures, facilitating more accurate and context-aware GED estimation (see Section 4.3). The *184 global semantic estimation* phase (detailed in Section 4.4) captures high-level differences between two KGs. First, a GNN is used to generate global representations for both input KGs. An NTN then processes these representations to model complex interactions from a global perspective. The resulting features are concatenated and fed into a Multi-Layer Perceptron (MLP) to produce the estimated GED. The learning objective is to minimize the discrepancy between the predicted and ground-truth GED. Additionally, to further guide training, we employ the Earth Mover's Distance (EMD) [60] between the global semantic features of the two KGs as an auxiliary loss, encouraging the model to align their overall semantic distributions better. Upon successful training, our `SEABED` model is capable of predicting the GED for new KGs unseen during the model training phase.

### 4.2 Initial embedding extraction

This stage contains two major steps, including initial embedding extraction, and graph-level representation.

**Node and edge embeddings.** The first step is carried out offline and consists of (1) obtaining KG pairs via sampling from a large KG for model training, (2) computing embeddings for atoms that

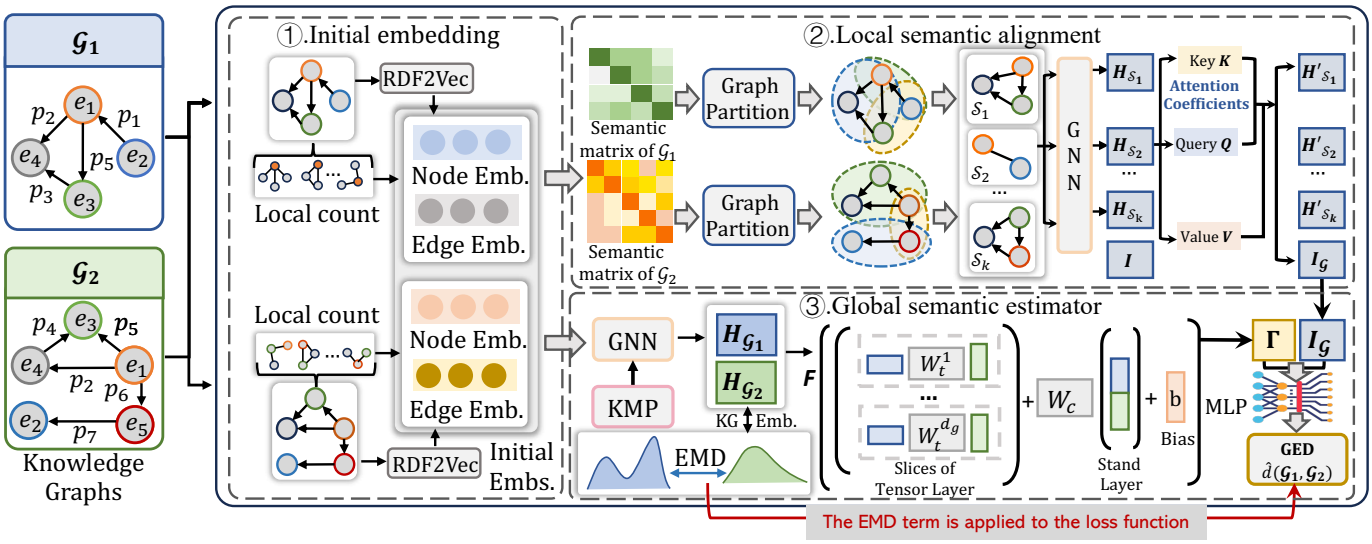



**Figure 4: The overview of SEABED.**

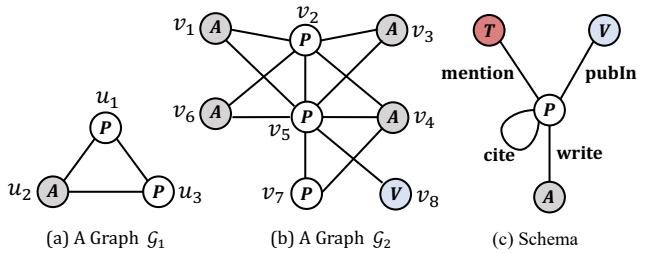
occur in the KG pairs, (3) building the featurization of each KG using (1) and (2). Specifically, SEABED first obtains a training set of KG pairs with their true GED from a given large KG, and SEABED calculates embeddings for each atom $x$ (node and predicate) using RDF2Vec [64]. Since as explained in Section 2.2, RDF2Vec embeddings encode relatedness and joint probability, we choose it as our embedding method. With RDF2Vec, nodes and predicates are treated as elements in the random walks, therefore, RDF2Vec is able to produce embeddings for all types of atoms in KG. These embeddings are then used as the node and edge embeddings for the KG pairs; that is, we obtain the initial embedding for each node and edge in KGs. We note that our method is not sensitive to the choice of initial embedding model. In our latter experiments, in addition to RDF2Vec, we also evaluate three widely used KGE models to demonstrate the robustness and generality of our approach.

**Local motif count-based features.** To further enhance these representations, we incorporate local subgraph count features, which capture the occurrence of characteristic structural patterns around each node. Motivated by the structural properties underlying GED, we leverage motif-based subgraph count information to enrich node representations. Specifically, we connect GED estimation with the concept of motif orbits. We first introduce the following definition:

*Definition 4.1 (Subgraph Isomorphism [58]).* A graph $\mathcal{G}_1$=($\mathcal{V}_1, \mathcal{E}_1$) is isomorphic to another graph $\mathcal{G}_2$=($\mathcal{V}_2, \mathcal{E}_2$) if there exists a bijective mapping $\Pi : \mathcal{V}_1 \to \mathcal{V}_2$ such that for all $u \in \mathcal{V}_1$, $\psi(u) = \psi(\Pi(u))$, and for all $u, v \in \mathcal{V}_1$, $(u, v) \in \mathcal{E}_1$ if and only if $(\Pi(u), \Pi(v)) \in \mathcal{E}_2$ and $\phi(u, v) = \phi(\Pi(u), \Pi(v))$, where $\Pi(u)$ denotes the vertex in $\mathcal{G}_2$ corresponding to vertex $u$ in $\mathcal{G}_1$ and $\psi(u)$ and $\phi(u, v)$ denote the labels of vertex $u$ and edge $(u, v)$, respectively.

Building upon this, we introduce a definition of graph orbit:

*Definition 4.2 (Graph Orbit).* Given two KGs, $\mathcal{G}_1$=($\mathcal{V}_1, \mathcal{E}_1$) and $\mathcal{G}_2$=($\mathcal{V}_2, \mathcal{E}_2$), let $u \in \mathcal{V}_1$ and $v \in \mathcal{V}_2$. We say that vertices $u$ and $v$ belong to the same graph orbit if there exists a subgraph isomorphism $\Pi : \mathcal{V}_1 \to \mathcal{V}_2$ between $\mathcal{G}_1$ and $\mathcal{G}_2$ such that $\Pi(u) = v$.




(a) A Graph $\mathcal{G}_1$ (b) A Graph $\mathcal{G}_2$ (c) Schema

**Figure 5: Illustrating the subgraph isomorphism.**

Example 2. *As shown in Figure 5, we consider two KGs, $\mathcal{G}_1$ = ($\mathcal{U}, \mathcal{E}_1$) and $\mathcal{G}_2$ = ($\mathcal{V}, \mathcal{E}_2$), together with their schema specifying edge types between entities. We observe that $\mathcal{G}_1$ is subgraph isomorphic to $\mathcal{G}_2$. Specifically, there exists a mapping $\Pi : \mathcal{U} \to \mathcal{V}$ such that $\Pi(u_1) = v_2$, $\Pi(u_2) = v_1$, and $\Pi(u_3) = v_5$. Under this mapping, both vertex and edge labels are preserved, i.e., $\psi(u_i) = \psi(\Pi(u_i))$ and $\phi(u_i, u_j) = \phi(\Pi(u_i), \Pi(u_j))$ for all $(u_i, u_j) \in \mathcal{E}_1$, where $\psi(\cdot)$ and $\phi(\cdot, \cdot)$ denote vertex and edge labels, respectively. For example, $\psi(u_1) = \psi(v_2) = P$, and the corresponding edge types are also consistent. Therefore, nodes mapped by $\Pi$ belong to the same orbit under the subgraph isomorphism.*

Given a motif $\mathcal{M}$ and a graph $\mathcal{G}$, a subgraph $\mathcal{G}' \subseteq \mathcal{G}$ is called a motif instance of $\mathcal{M}$ if and only if the motif $\mathcal{M}$ is subgraph isomorphic to subgraph $\mathcal{G}'$. Based on this notion, we establish the following lemma, which connects local subgraph counts with GED computation.

Lemma 4.3. *Given two KGs $\mathcal{G}_1$=($\mathcal{V}_1, \mathcal{E}_1$) and $\mathcal{G}_2$=($\mathcal{V}_2, \mathcal{E}_2$), if GED between the two KGs is 0, then for any motif $\mathcal{M}$, if $u \in \mathcal{V}_1$ and $v \in \mathcal{V}_2$ are in the same graph orbit, we have $\Psi_{\mathcal{M}}(u, \mathcal{G}_1) = \Psi_{\mathcal{M}}(v, \mathcal{G}_2)$, where $\Psi_{\mathcal{M}}(u, \mathcal{G})$ denotes the number of motif instances in $\mathcal{G}$ containing $u$.*

The above lemma indicates that local motif count captures essential local structural information relevant to GED. While identical local motif count vectors do not guarantee subgraph isomorphism,

the converse does hold: if two KGs are isomorphic (i.e., GED = 0), then every pair of corresponding nodes must share the same local motif count vectors. Therefore, we leverage the local motif count vector of each node as a local structural feature. Specifically, for each node, we compute its frequency in five representative motif structures [85, 87] (see our technical report [88]), disregarding label information. This results in a five-dimensional count vector for each node, where each entry reflects the frequency of a particular motif type. Finally, we concatenate the local motif count vector with the initial node embedding (obtained via RDF2Vec) to form the final node representation. This approach effectively integrates both semantic and local structural features of KG, providing a richer foundation for the GED computation.

## 4.3 Local semantic alignment

In this subsection, we describe how SEABED achieves local semantic alignment. Specifically, it consists of two steps: (1) semantics-aware KG partition, and (2) subgraph-based self-attention mechanism.

**Semantics-aware KG partition.** Given a KG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we first construct a semantic similarity matrix $\mathcal{A} \in \mathbb{R}^{n \times n}$, where each entry of the matrix is defined as the cosine similarity between the embeddings of two nodes:

$$\mathcal{A}_{ij} = \frac{h_i \cdot h_j^\top}{\|h_i\| \|h_j\|}, \tag{3}$$

where $h_i$ and $h_j$ are the embeddings of nodes $x_i$ and $x_j$, respectively.

Intuitively, our goal is to assign nodes with high semantic similarity to the same subgraph, so that each subgraph forms a coherent semantic unit. That is, an ideal partitioning method should maximize the sum of node similarities within each cluster, while minimizing the total similarity of nodes that are cut between different clusters. This ensures that subgraphs exhibit strong internal semantic coherence and clear separation from each other. To achieve this, we adapt the widely used METIS [32] algorithm for semantics-aware partitioning. Specifically, we encode the pairwise semantic similarities between nodes as edge weights in the input graph. METIS then partitions the graph in a way that high-similarity nodes are more likely to be grouped together, resulting in subgraphs that better reflect the underlying semantic structure of the KG.

Using this method, we partition each KG into $k/2$ sub-KGs, resulting in a set of subgraphs $\{\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_k\}$, each with high internal semantic similarity. To mitigate the potential loss of edge information and semantic discontinuity caused by non-overlapping partitions, we enforce a one-hop overlap between any two adjacent sub-KGs, ensuring smoother semantic transitions and better connectivity across subgraphs.

**Subgraph-based self-attention.** After obtaining $k$ sub-KGs, we apply a GNN to aggregate features within each expanded subgraph. Specifically, for each sub-KG $\mathcal{S}_i$, this yields an embedding $H_{\mathcal{S}_i} \in \mathbb{R}^s$ that captures both semantic and structural information. The main challenge is to effectively identify, align, and compare semantically entangled subgraphs between two KGs. Not all sub-KGs are equally informative for GED estimation; some encapsulate critical semantic or structural patterns that reflect meaningful differences between graphs, while others may be redundant or irrelevant. Therefore, we require an adaptive alignment mechanism that can (i) emphasize semantically relevant subgraph correspondences, (ii) suppress noisy

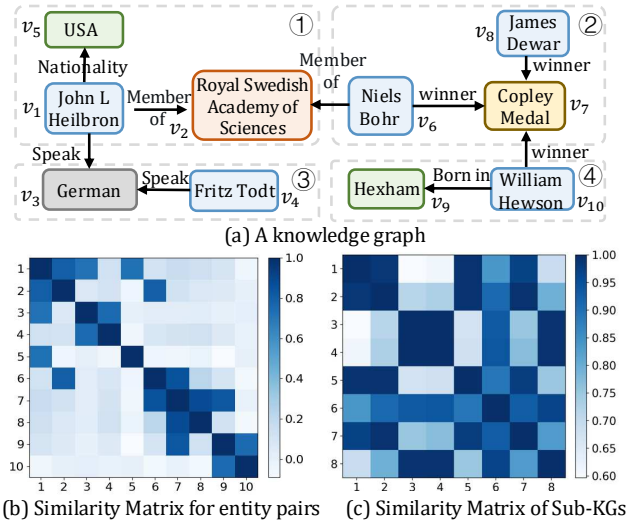



**Figure 6: Illustrating the local semantic alignment.**

or spurious matches, and (iii) flexibly handle variations in sub-KG size and semantics across graphs.

Although the above task seems to be very difficult, we borrow the successful experience of applying the self-attention mechanism [71] to fulfill such goals. The self-attention mechanism allows models to learn to concentrate on relevant aspects of input data while disregarding the irrelevant, thereby mimicking human cognitive attention processes.

In particular, let $X = \begin{bmatrix} x_1, & x_2, & \cdots, & x_n \end{bmatrix} \in \mathbb{R}^d$ be input data with $n$ columns, where each column $x_i$ represents an input token (e.g., a word embedding) with the same dimension $d$. The self-attention mechanism transforms this input matrix into three matrices: Key ($K$), Query ($Q$), and Value ($V$) through learned weight matrices: $Q = X\mathbf{W}_Q, K = X\mathbf{W}_K$, and $V = X\mathbf{W}_V$, where $\mathbf{W}_Q \in \mathbb{R}^{d \times d_q}$, $\mathbf{W}_K \in \mathbb{R}^{d \times d_k}$ and $\mathbf{W}_V \in \mathbb{R}^{d \times d_v}$ are learnable matrices for queries, keys, and values, respectively, and $d_q$=$d_k$ usually. Denote $q_i$, $k_i$, and $v_i$ as the $i$-th row of the matrices $Q$, $K$, and $V$, respectively. The attention coefficients $\alpha_{i,j}$, which reflect the importance from $x_j$ to $x_i$, are computed as:

$$\alpha_{i,j} = \frac{\exp\left(\frac{q_i \cdot k_j^\top}{\sqrt{d_k}}\right)}{\sum_{l=1}^{n} \exp\left(\frac{q_i \cdot k_l^\top}{\sqrt{d_k}}\right)}, \tag{4}$$

where $q_i \cdot k_j^\top$ denotes the correlation coefficient between query $i$ and key $j$ and $\sqrt{d_k}$ is a normalization factor. Based on these scores, we can obtain a number of output tokens $y_1, y_2, \cdots, y_n$, where $y_i$=$\sum_j \alpha_{i,j} v_j$.

Let matrix $Y = \begin{bmatrix} y_1, & y_2, & \cdots, & y_n \end{bmatrix}^\top \in \mathbb{R}^{n \times d_v}$. The whole self-attention process can be formulated as follows:

$$Y = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V. \tag{5}$$

The self-attention mechanism has two advantages: First, it could condense the information of all input vectors to the outputs. In the NLP domain, this is applied to compress the semantics of contextual

words, phrases, and sentences. For our GED estimation problem, it could be applied to compress the $k$ sub-KGs into a more concise, high-level representation. Second, for different input embedding vectors, the attention coefficients $\alpha_{i,j}$ are different. This allows the outputs to adaptively focus on different parts of the sub-KGs. In the NLP task, this is useful to resolve the semantics of demonstrative pronouns. For our GED estimation problem, this mechanism enables us to: (1) process the highly and weakly correlated sub-KGs in different ways. (2) align the sub-KGs with different weights for different semantic and structural characters in KGs. This way, the learned model could be applicable to different settings.

Thus, the self-attention module fully satisfies the requirement of capturing local semantic information, allowing SEABED to identify semantic differences between subgraphs effectively. To this end, we incorporate self-attention as a core component of our model. Specifically, we treat each sub-KG as a token—similar to word tokens in NLP [71] or image patches in computer vision [15]. However, two key challenges arise: **Challenge 1**: How to effectively encode the position of each sub-KG (token)? **Challenge 2**: How to aggregate information across tokens to capture their semantic relationships?

For **Challenge 1**, we introduce a novel subgraph position embedding strategy. Unlike sequences, general graphs lack a fixed ordering between subgraphs; instead, their relationships are defined by the strength of topological connections. To capture the relative positions between sub-KGs, we define a position matrix $P \in \mathbb{R}^{k\times k}$, where each entry $P_{i,j}$ represents the positional relationship between sub-KG $\mathcal{S}_i$ and sub-KG $\mathcal{S}_j$:

$$P_{i,j} = \mathtt{cut}(\mathcal{S}_i, \mathcal{S}_j), \tag{6}$$

where $\mathtt{cut}(\mathcal{S}_i, \mathcal{S}_j) = \sum_{x_i\in\mathcal{S}_i}\sum_{x_j\in\mathcal{S}_j}\mathbb{1}[e_{i,j} \in \mathcal{E}]$ quantifies the number of connecting edges between sub-KG $\mathcal{S}_i$ and sub-KG $\mathcal{S}_j$ For each sub-KG, we use $P_i \in \mathbb{R}^{k\times 1}$, the $i$-th column of the position matrix, to capture its relative position to all other sub-KGs. The positional encoding is then integrated into the subgraph features via:

$$H_{\mathcal{S}_i} = H_{\mathcal{S}_i} + W_P \cdot P_i, \tag{7}$$

where $W_P \in \mathbb{R}^{d\times k}$ is a learnable parameter.

To address **Challenge 2**, we randomly initialize a special embedding vector $\mathcal{I} \in \mathbb{I}^d$ as input. $\mathcal{I}$ would be fed into the subsequent modules with the other embedding vectors together. It serves to amalgamate the information of other embedding vectors together, akin to the CLS token in NLP tasks [71] or the virtual node in GNN [23]. To this end, we can employ the self-attention module to capture semantic dependencies and interactions among sub-KGs. In particular, let $\boldsymbol{H}_{\mathcal{S}} = [H_{\mathcal{S}_1}, H_{\mathcal{S}_2}, \ldots, H_{\mathcal{S}_k}, \mathcal{I}]$. Specifically, in our method, the features of the $k$ subgraphs $H_S$, are used as queries ($Q$), keys ($K$), and values ($V$) for the self-attention computation,

$$\boldsymbol{H}'_S = \mathrm{softmax}\left(\frac{(\boldsymbol{H}_S\mathbf{W}_Q)(\boldsymbol{H}_S\mathbf{W}_K)^\top}{\sqrt{d_k}}\right)\cdot(\boldsymbol{H}_S\mathbf{W}_V), \tag{8}$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are learnable weight matrices.

During the computation, the semantic information of the sub-KGs is fused via the self-attention mechanism, enabling the model to automatically adjust weights based on the semantic relationships between subgraphs and focus on the most critical parts for GED estimation. Finally, we extract the output vector in $\boldsymbol{H}'_S$ corresponding to $\mathcal{I}_{\mathcal{G}}$ as the final representation. This vector can be regarded as a high-order summarization of both the outputs and the input features. As a result, we only retain this condensed representation for use in the next stage.

To better illustrate how our partitioning strategy captures semantic entanglement, we provide an example in Figure 6. Specifically, Figure 6(a) presents a KG with 10 entities describing relationships surrounding the Nobel Prize. Figure 6(b) shows the semantic similarity matrix of all entity pairs, where darker colors indicate higher similarity. Based on this matrix, the KG is partitioned into four sub-KGs (the index of each sub-KG is shown in the upper-right corner). We observe that entities with strong semantic relevance are naturally grouped together, and each sub-KG forms a coherent semantic unit. Figure 6(c) further shows the similarity matrix between the KG in Figure 6(a) and another KG when computing the GED between them, where both KGs are partitioned into four sub-KGs (eight sub-KGs in total). This matrix captures the pairwise similarities among all sub-KG pairs and serves as the basis for SEABED to estimate the GED. Consequently, SEABED assigns larger attention weights to these coherent regions, enabling more accurate alignment of the local semantic units across the two KGs.

## 4.4 Global semantic estimator

In this section, we introduce the implementation of the Global Semantic Estimator and discuss how it integrates both global and local information to predict the GED.

**Graph-level representation.** Figure 4 shows the SEABED architecture that consists of a GNN that implements Message Passing Layers, and an NTN to capture the interaction of two KGs. To design a KG-native graph representation method, we require a Message Passing strategy that models both edge labels and their directions. This is because, unlike graphs that only contain node labels and are undirected, KGs not only carry rich semantic information in their edges (i.e., predicates) but also are directed. Hence, instead of using a standard Message-Passing function, we design a KG-native message-passing (KMP) function, specifically designed to handle the semantic information embedded in both the nodes and edges of the KG. The KMP is extended by the Message Passing function from GINEConv [27]:

$$\begin{aligned} h_i^{(k)} = \mathrm{MLP}^{(k)}\Big((1+\eta)\cdot h_i^{(k-1)} + \\ \sum_{x_j\in N^-(x_i,\mathcal{G})} F\left(\mathbf{W}_g^-\left(h_i^{(k-1)} \parallel h_{i,j} \parallel h_j^{(k-1)}\right)\right) + \\ \sum_{x_j\in N^+(x_i,\mathcal{G})} F\left(\mathbf{W}_g^+\left(h_j^{(k-1)} \parallel h_{j,i} \parallel h_i^{(k-1)}\right)\right)\Big). \end{aligned} \tag{9}$$

This function makes a case distinction between incoming and outgoing edges, differentiated by $N^+$ and $N^-$, respectively. Instead of aggregating the node features $h_i$, $h_j$ and edge features $h_{i,j}$ through summation, the function concatenates them ($\parallel$) and linearly projects them to the same dimensionality as $h_i$ using the matrices $\mathbf{W}_g^+$, and $\mathbf{W}_g^-$. Here, $F$ is a ReLU activation function, and we utilize two different matrices $\mathbf{W}_g^+$ and $\mathbf{W}_g^-$ to capture the direction of edges. Besides, for two KGs, we also do not need two matrices $\mathbf{W}_{g_1}$ and $\mathbf{W}_{g_2}$, since parameter sharing is a commonly used way in the machine learning area [4, 5, 54, 61] to reduce the model size

and improve the training efficiency. Lastly, the $\text{MLP}^{(k)}$ receives as input sums of triple pattern representations.

In a preliminary study, we found that using more layers does not improve the model's performance. Possible explanations for this are that (1) the GED calculation typically focuses on small KGs, so two steps of message-passing are sufficient, and (2) fewer layers avoid the over-smoothing problem [11], where node representations become increasingly similar, which deteriorates model performance. Hence, we use the 2-layer GNN in SEABED.

After the KG nodes have been transformed by the two Message Passing Layers, they need to be aggregated into a fixed-length vector. For that, SEABED applies a Global Sum Aggregation to them. That is, the representations of the nodes are summed up dimension-wise, resulting in a single vector $H_{\mathcal{G}}$ with the same dimensionality as the node embeddings:

$$H_{\mathcal{G}} = \sum_{i}^{n} h_i^{(k=2)}, \tag{10}$$

where $H_{\mathcal{G}} \in \mathbb{R}^d$ is a $d$-dimensional embedding to represent the high-level information of $\mathcal{G}$.

**NTN interaction module.** Given two KGs, $\mathcal{G}_1$ and $\mathcal{G}_2$, the GNN model of SEABED generates two KG embeddings, $H_{\mathcal{G}_1}$ and $H_{\mathcal{G}_2}$, respectively. Recall that GED measures the similarity between two KGs. Here, to capture the intricate relationships between these embeddings, we employ an NTN [63], similar to existing methods, which operates as follows:

$$\Gamma = F\left(H_{\mathcal{G}_1}^{\top} \mathbf{W}_t^{[1\cdots d_g]} H_{\mathcal{G}_2} + \mathbf{W}_c(H_{\mathcal{G}_1} \parallel H_{\mathcal{G}_2}) + \mathbf{b}\right), \tag{11}$$

where $\mathbf{W}_t \in \mathbb{R}^{d\times d\times d_g}$, $\mathbf{W}_c \in \mathbb{R}^{d_g\times 2d}$ and $\mathbf{b} \in \mathbb{R}^{d_g}$ are learnable matrices, $\parallel$ denotes the concatenation of two vectors, and $F$ is a ReLU activation function. The hyperparameter $d_g$ is used to control the dimensionality of the embeddings after interaction. The NTN allows the SEABED to capture the structural differences between two KGs from a global perspective. In addition, the matrix $\mathbf{W}_c$ acts as a channel attention mechanism, assigning different weights to each dimension of the embeddings $H_{\mathcal{G}_1}$ and $H_{\mathcal{G}_2}$. This allows the model to selectively focus on the most important features of both KGs, enhancing its ability to capture critical structural information.

**Multi-scale semantic fusion.** In SEABED, we improve the prediction accuracy of the GED by integrating both local and global information. Specifically, we concatenate the local semantic vector $\mathcal{I}_{\mathcal{G}}$ and the global semantic vector $\mathbf{\Gamma}$, forming a comprehensive feature vector $\mathbf{U} = [\mathbf{\Gamma} \parallel \mathcal{I}_{\mathcal{G}}]$. This concatenation preserves the independence of local and global information while enabling the model to capture their inter-dependencies, thereby more accurately characterizing the differences between the graphs. Next, we use a Multi-Layer Perceptron (MLP) to process the concatenated feature vector and predict the graph edit distance $\widehat{d}(\mathcal{G}_1, \mathcal{G}_2) = \text{MLP}(\mathbf{U})$.

## 4.5 Model training

In SEABED, we not only employ supervised learning to train the model – minimizing the mean squared error (MSE) between the predicted GED $\widehat{d}(\mathcal{G}_1, \mathcal{G}_2)$ and the true GED $d(\mathcal{G}_1, \mathcal{G}_2)$ – but also introduce the Earth Mover's Distance (EMD) as an auxiliary loss term. EMD serves as a key regularization for global semantic alignment, encouraging the learned representations of the two KGs to capture their overall semantic similarity. Specifically, the MSE loss for GED prediction is defined as:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{|D|} \sum_{(i,j)\in D} \left(\widehat{d}(\mathcal{G}_i, \mathcal{G}_j) - d(\mathcal{G}_i, \mathcal{G}_j)\right)^2, \tag{12}$$

where $D$ is the set of KG pairs in the training dataset, $\widehat{d}(\mathcal{G}_i, \mathcal{G}_j)$ is the predicted GED, and $d(\mathcal{G}_i, \mathcal{G}_j)$ is the true GED.

On the other hand, EMD measures the difference between two KGs in the global feature space, enhancing the model's ability to capture global information. The EMD is computed as:

$$\mathcal{L}_{\text{EMD}} = \int_{-\infty}^{+\infty} \left|\hat{F}_{H_{\mathcal{G}_1}}(x) - \hat{F}_{H_{\mathcal{G}_2}}(x)\right| dx, \tag{13}$$

where $\hat{F}_{H_{\mathcal{G}_1}}$ and $\hat{F}_{H_{\mathcal{G}_2}}$ are the empirical cumulative distribution functions (CDFs) of the graph-level representations $H_{\mathcal{G}1}$ and $H_{\mathcal{G}_2}$, respectively. Intuitively, this loss measures the overall discrepancy between the two feature distributions. For the common case where both empirical distributions assign equal weights to all dimensions, the EMD admits a simple closed-form:

$$\mathcal{L}_{\text{EMD}} = \frac{1}{d} \sum_{i=1}^{d} \left|\xi_{H_{\mathcal{G}_1}}(i) - \xi_{H_{\mathcal{G}_2}}(i)\right|, \tag{14}$$

where $\xi_{H_{\mathcal{G}_1}}$ and $\xi_{H_{\mathcal{G}_2}}$ are the feature values of $H_{\mathcal{G}1}$ and $H_{\mathcal{G}_2}$ sorted in ascending order, and $d$ denotes the dimensionality of the graph-level representations. The total loss function combines the MSE loss and the EMD loss term:

$$\mathcal{L}_{\texttt{Total}} = \mathcal{L}_{\text{MSE}} + \lambda \cdot \mathcal{L}_{\text{EMD}}, \tag{15}$$

where $\lambda$ is used to control the weight of the EMD loss term.

# 5 EXPERIMENTS

We now present the experimental results. Section 5.1 discusses the setup. We discuss the results in Sections 5.2 and 5.3.

## 5.1 Setup

**Table 1: Meta KG characteristics.**

| Name | Triples | Entities | Predicates | Classes |
|---|---|---|---|---|
| SWDF | 242,256 | 76,711 | 170 | 118 |
| LUBM | 2,688,849 | 664,048 | 18 | 15 |
| YAGO | 58M | 13M | 92 | 189K |
| WIKIDATA | 21M | 5M | 872 | 19K |

**Table 2: Statistics of KG sets.**

| Name | # KGs | $\lvert\overline{V}\rvert$ | $\lvert\overline{E}\rvert$ | $\lvert V\rvert_{\max}$ | $\lvert E\rvert_{\max}$ | # pairs |
|---|---|---|---|---|---|---|
| SWDF | 1000 | 7 | 8 | 10 | 20 | 50000 |
| LUBM | 1000 | 7 | 6 | 10 | 20 | 50000 |
| YAGO | 1000 | 19 | 21 | 25 | 40 | 30000 |
| WIKIDATA | 1000 | 25 | 27 | 30 | 78 | 50000 |

*5.1.1 Datasets.* We use four real-world knowledge graphs (KGs) as meta KG – SWDF [48], LUBM [22], YAGO [65], and WIKIDATA [76] – to construct our training and test sets. The statistics of these KGs are reported in Table 1. The detailed descriptions of each dataset are provided in our technical report [88]. We use the aforementioned KGs as meta KGs and create KG pairs by generating smaller KGs from them. To do this, we employ the random walk, which is a commonly used method for extracting small, representative subgraphs from large graphs [7, 66, 87]. Table 2 presents the statistics of the KGs generated from meta KGs. Specifically, # KGs denotes the number of KGs; $|\overline{V}|$ and $|\overline{E}|$ represent the average number of vertices and edges; $|V|_{\max}$ and $|E|_{\max}$ are the maximum number of vertices and edges, respectively; and # pairs indicates the number of KGs pairs used for training our SEABED to estimate the GEDs.

*5.1.2 Ground-truth GED Generation.* Since all KGs in SWDF and LUBM have no more than 10 nodes, we simply use the exact algorithm to generate the ground-truth data on these KGs. However, due to the NP-hardness of GED computation, the ground-truth data of YAGO and WIKIDATA is usually generated in a sub-optimal manner instead. Hence, we follow the existing works [4, 54] that use a simple way to synthesize KG pairs in a given data set. Specifically, for each KG $\mathcal{G}$, we can randomly apply $\Delta$ graph edit operations on it and get a synthetic KG $\mathcal{G}'$. Notably, when constructing a new KG, we perform fact validation to ensure that all encoded facts are correct and semantically consistent. Note that we have $\text{GED}(\mathcal{G}, \mathcal{G}') \leq \Delta$. If the size of $\mathcal{G}$ is large enough w.r.t. $\Delta$ and these operations edit distinct nodes/edges, the probability that $\text{GED}(\mathcal{G}, \mathcal{G}') < \Delta$ is quite low. In this case, $\Delta$ can be approximately regarded as the ground-truth GED of this synthetic graph pair $(\mathcal{G}, \mathcal{G}')$. For the four datasets SWDF, LUBM, YAGO, and WIKIDATA, their GED values are randomly distributed in $[1, 20]$, $[1, 17]$, $[5, 16]$, and $[5, 16]$, respectively. We acknowledge the limitation that this dataset is generated through simulation rather than using true KG evolution data, which may affect the performance of our model in real-world scenarios.

*5.1.3 Data Partition.* We divide the KG pairs for the four datasets into training, validation, and test sets using a 6:2:2 ratio, following this classical setting [5, 54, 83].

*5.1.4 Competitors.* Existing methods for GED computing can be divided into two categories: learning-based and traditional. For the former type, we mainly compare our method with five approaches: SimGNN [5], TaGSim [4], GPN [83], GEDGNN [54], and GEDIOT [12]. Besides, we also compare our method with the two representative traditional methods: Greedy [36], and Noah [83]. The detailed descriptions of each method are shown in our technical report [88].

*5.1.5 Detailed Setup.* **Experimental setup.** We implement all the algorithms in Python with PyTorch, and run experiments on a machine having an Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz and NVIDIA GeForce RTX 3090 GPU, with Ubuntu installed. If an algorithm cannot finish in 48 hours, we mark its running time as "—" in the table. We provide the detailed hyperparameter settings of all experiments in our technical report [88].

**Evaluation metrics.** To demonstrate the superiority of SEABED, we evaluate the performance of our model using the following metrics that are commonly used in the previous works [4, 5, 54, 83]:

- **GED prediction accuracy**: We use two widely used metrics Mean Absolute Error (MAE) and accuracy to measure the performance of our model. Specifically, given the true GED value $d$ and the predicted GED value $\widehat{d}$ for each test KG pair, the MAE is defined as $|d - \widehat{d}|$, and accuracy is defined as the proportion of test KG pairs for which $\lceil \widehat{d} \rceil = d$.
- **Ranking metrics**: For each test KG $\mathcal{G}$, we have 30 KGs that form the testing set. These KGs can be ranked based on their true GED values relative to the test KG $\mathcal{G}$. Therefore, we use Spearman's rank correlation coefficient $\gamma$, Kendall's rank correlation coefficient $\tau$, and precision at ranks 5, 10, 15, and 20 (denoted as $p$@5, $p$@10, $p$@15 and $p$@20) as evaluation metrics.

## 5.2 Comparison with the existing methods

In this section, we present the metric values calculated by comparing the ground-truth GEDs with those obtained from different GED estimation methods.

**1. GED precision accuracy.** As shown in Table 3, we compare the MAE and Accuracy values for all GED estimation methods. Specifically, each method is run five times independently, and we report the average performance across all runs for each evaluation metric. In Table 3, each result is presented in the form of $\mu \pm \sigma$, where $\mu$ denotes the mean value over five independent runs and $\sigma$ represents the corresponding standard deviation, computed as: $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$, where $N$ is the total number of runs and $x_i$ denotes the result of the $i$-th run. Based on the results, we can make the following observations and analysis: (1) For the best results (i.e., $\mu + \sigma$), SEABED significantly outperforms all existing methods in terms of MAE and Accuracy. For example, the MAE is reduced by up to 66.7%, while accuracy improves by up to 70.5%, demonstrating the effectiveness of SEABED in estimating GED. Even when considering the average performance (i.e., $\mu$), SEABED still achieves up to a 73.8% reduction in MAE and a 103% improvement in accuracy, demonstrating both the effectiveness and robustness of our approach. This is because SEABED is a KG-native model that not only employs a novel local semantic alignment strategy to capture regional structural and semantic differences, but also introduces EMD to align global semantic distributions between knowledge graphs. (2) For larger KGs, such as YAGO and WIKIDATA, SEABED shows more significant improvements compared to smaller KGs. For instance, when compared to the second-best algorithm, the MAE decreases from 0.391 to 0.212 on WIKIDATA, while accuracy increases from 57.6% to 98.2% on YAGO. This is because larger KGs typically contain a vast number of different types of entities and predicates, which makes existing learning-based methods struggle to capture the key structural and semantic differences between the two KGs. Besides, we also compute the $p$-values [79] (a well-known statistical significance measure) between GEDIOT and SEABED to quantify the performance advantage of our method. For both MAE and Accuracy, the $p$-values are below 0.05 across all datasets, indicating that the improvements achieved by SEABED are statistically significant. Moreover, for Accuracy, the $p$-values are below 0.01, suggesting a highly significant improvement. Due to the space limitation, the details are presented in our technical report [88].

**Table 3: Performance of all GED computation methods, where shaded areas indicate the best results and orange marks the second-best results.**

| SWDF | GED | | Ranking | | | | | |
|---|---|---|---|---|---|---|---|---|
| | MAE | Accuracy(%) | $\gamma$ | $\tau$ | $p$@5(%) | $p$@10(%) | $p$@15(%) | $p$@20(%) |
| SimGNN | 0.799±0.015 | 41.39±0.80 | 0.823±0.002 | 0.692±0.003 | 41.74±1.00 | 61.83±0.20 | 72.94±0.37 | 79.36±0.14 |
| TaGSim | 1.334±0.023 | 28.68±2.80 | 0.770±0.023 | 0.642±0.021 | 43.96±3.08 | 63.23±11.17 | 67.20±1.78 | 75.29±1.53 |
| GEDGNN | 0.873±0.035 | 34.72±1.29 | 0.865±0.003 | 0.741±0.002 | 44.25±1.46 | 62.38±0.54 | 72.91±0.48 | 79.76±0.24 |
| GPN | 1.011±0.152 | 31.21±1.95 | 0.844±0.035 | 0.719±0.036 | 45.00±1.51 | 60.92±1.37 | 71.92±2.31 | 78.68±1.69 |
| GEDIOT | 0.771±0.068 | 40.57±2.29 | **0.906±0.014** | **0.792±0.019** | **53.19±1.35** | **68.04±1.58** | **78.08±1.65** | **83.58±1.16** |
| Greedy | 4.034±0.000 | 8.90±0.00 | 0.262±0.000 | 0.321±0.000 | 24.20±0.00 | 36.30±0.00 | 47.50±0.00 | 56.70±0.00 |
| Noah | — | — | — | — | — | — | — | — |
| SEABED | **0.675±0.007** | **47.40±0.38** | 0.886±0.002 | 0.769±0.003 | 52.81±0.89 | 67.32±0.49 | 77.27±0.50 | 82.52±0.40 |
| **LUBM** | GED | | Ranking | | | | | |
| | MAE | Accuracy(%) | $\gamma$ | $\tau$ | $p$@5(%) | $p$@10(%) | $p$@15(%) | $p$@20(%) |
| SimGNN | 1.113±0.269 | 28.76±4.96 | 0.750±0.049 | 0.614±0.045 | 36.04±8.06 | 54.42±7.44 | 66.05±5.20 | 74.23±3.13 |
| TaGSim | 1.182±0.018 | 49.73±0.74 | 0.818±0.005 | 0.699±0.005 | 47.34±1.98 | 62.56±0.79 | 70.26±0.84 | 76.14±0.78 |
| GEDGNN | 0.701±0.026 | 43.84±2.60 | 0.887±0.002 | 0.762±0.014 | 48.72±1.15 | 63.35±0.85 | 73.84±0.79 | 81.04±0.27 |
| GPN | 0.877±0.125 | 35.50±5.18 | 0.816±0.052 | 0.686±0.061 | 42.58±3.95 | 59.98±2.28 | 70.81±2.68 | 77.71±2.39 |
| GEDIOT | 0.702±0.017 | 44.35±1.77 | 0.882±0.003 | 0.762±0.003 | 46.18±1.86 | 63.17±0.89 | 73.80±0.48 | 80.62±0.19 |
| Greedy | 1.988±0.000 | 11.80±0.00 | 0.475±0.000 | 0.405±0.000 | 33.70±0.00 | 49.20±0.00 | 61.50±0.00 | 68.40±0.00 |
| Noah | — | — | — | — | — | — | — | — |
| SEABED | **0.590±0.011** | **52.29±1.05** | **0.925±0.002** | **0.824±0.003** | **66.73±1.14** | **74.48±0.57** | **80.72±0.47** | **85.23±0.23** |
| **YAGO** | GED | | Ranking | | | | | |
| | MAE | Accuracy(%) | $\gamma$ | $\tau$ | $p$@5(%) | $p$@10(%) | $p$@15(%) | $p$@20(%) |
| SimGNN | 0.831±0.021 | 31.93±0.99 | 0.922±0.000 | 0.799±0.000 | 70.81±0.62 | 85.67±0.18 | 89.78±0.14 | 92.37±0.06 |
| TaGSim | 2.285±0.935 | 13.04±3.38 | 0.889±0.045 | 0.768±0.052 | 70.89±5.04 | 83.82±3.65 | 88.59±2.88 | 87.68±7.39 |
| GEDGNN | 0.820±0.099 | 36.42±4.15 | 0.964±0.005 | 0.882±0.011 | 80.13±2.10 | 87.82±3.87 | 93.33±0.52 | 95.04±0.55 |
| GPN | 0.898±0.115 | 34.76±4.30 | 0.958±0.015 | 0.871±0.034 | 80.27±3.66 | 89.30±1.78 | 92.62±1.57 | 94.26±1.13 |
| GEDIOT | 0.678±0.227 | 47.01±11.12 | 0.973±0.020 | 0.909±0.048 | 82.56±4.93 | 91.29±2.46 | 94.14±2.05 | 95.62±1.64 |
| Greedy | 14.936±0.000 | 0.00±0.00 | 0.833±0.000 | 0.698±0.000 | 60.69±0.00 | 79.40±0.00 | 85.93±0.00 | 90.32±0.00 |
| Noah | — | — | — | — | — | — | — | — |
| SEABED | **0.177±0.032** | **95.78±2.38** | **0.991±0.000** | **0.951±0.000** | **87.84±0.43** | **93.68±0.12** | **95.70±0.08** | **97.06±0.07** |
| **WIKIDATA** | GED | | Ranking | | | | | |
| | MAE | Accuracy(%) | $\gamma$ | $\tau$ | $p$@5(%) | $p$@10(%) | $p$@15(%) | $p$@20(%) |
| SimGNN | 0.868±0.018 | 31.08±1.21 | 0.921±0.001 | 0.793±0.002 | 54.35±1.29 | 76.82±0.17 | 83.71±0.23 | 88.02±0.18 |
| TaGSim | 3.127±1.133 | 9.97±3.86 | 0.832±0.084 | 0.683±0.093 | 50.90±6.74 | 71.55±3.86 | 76.52±6.35 | 81.69±5.48 |
| GEDGNN | 0.866±0.121 | 36.10±6.13 | 0.962±0.016 | 0.892±0.015 | 67.87±2.01 | 86.29±1.32 | 89.36±0.71 | 92.85±0.49 |
| GPN | 1.219±0.226 | 24.85±4.57 | 0.959±0.021 | 0.872±0.045 | 66.03±4.79 | 84.20±4.39 | 87.93±3.14 | 91.71±2.24 |
| GEDIOT | 0.697±0.272 | 45.59±15.15 | 0.979±0.013 | 0.916±0.033 | 69.73±4.04 | 87.88±2.88 | 90.71±2.20 | 94.05±1.64 |
| Greedy | 21.742±0.000 | 0.00±0.00 | 0.831±0.000 | 0.695±0.000 | 49.20±0.00 | 65.19±0.00 | 75.99±0.00 | 82.45±0.00 |
| Noah | — | — | — | — | — | — | — | — |
| SEABED | **0.242±0.030** | **89.48±1.95** | **0.992±0.000** | **0.948±0.000** | **73.89±0.71** | **90.90±1.10** | **92.60±0.04** | **95.48±0.16** |

**2. Ranking metrics.** As for the ranking metrics, SEABED almost achieves the best performance, especially on the rank correlation coefficient $\gamma$. For example, on the LUBM dataset, our SEABED achieves a $\gamma$ score at 0.928, surpassing GEDIOT (0.893) by 3.91% and SimGNN (0.881) by 5.33%, respectively. For the two larger KGs, we can see that SEABED always performs better than the competitors, in terms of precision and coefficient values. The above results demonstrate that SEABED can be used to effectively identify similar KGs for a given KG. Besides, traditional algorithms usually underperform compared to learning-based algorithms due to their lower accuracy in GED prediction. In general, more accurate GED predictions lead to better performance in the above downstream applications.

## 5.3 Detailed analysis of SEABED

In this subsection, we first conduct several ablation studies and then present the detailed analysis for SEABED.

**1.Ablation studies.** To evaluate the effectiveness of our proposed key component in SEABED, we design four variants of the SEABED: (1) `w/o Local`: Remove the local semantic alignment module from SEABED. (2) `w/o Motif-Count`: Exclude the local motif count feature from each node's initial embedding. (3) `w/o EMD`: Omit the Earth Mover's Distance (EMD) as an auxiliary loss function in the training stage, and (4) `w/o Pos.-Emb.`: Remove positional encoding from the self-attention SEABED. We then compare these variants with SEABED and present the results in Table 4. Based on the results, we can make the following observation: (1) The attention-based local semantic alignment module effectively detects subgraph differences between KGs, resulting in an average MAE reduction of 21.72% and an accuracy improvement of 7.27%. (2) The local count information helps the model learn the KG structure, thereby retaining the most valuable features for GED estimation, leading to an average MAE reduction of 13.54% and an accuracy improvement of

**Table 4: Comparative results of SEABED and its variants.**

| SWDF | GED | | Ranking | | | | | |
|---|---|---|---|---|---|---|---|---|
| | MAE | Accuracy(%) | $\gamma$ | $\tau$ | $p$@5(%) | $p$@10(%) | $p$@15(%) | $p$@20(%) |
| w/o Local | 0.707±0.007 | 44.16±2.31 | 0.879±0.003 | 0.759±0.004 | 50.96±1.67 | 66.93±0.54 | 76.79±0.50 | 81.96±0.42 |
| w/o Motif-Count | 0.709±0.005 | 45.71±0.22 | 0.878±0.003 | 0.756±0.003 | 50.33±0.48 | 66.43±1.43 | 76.95±0.29 | 82.06±0.21 |
| w/o EMD | 0.689±0.013 | 46.74±1.17 | 0.883±0.005 | 0.766±0.007 | 51.61±0.95 | 67.28±0.52 | 77.25±0.45 | 82.32±0.39 |
| w/o Pos.-Emb. | 0.681±0.007 | 47.17±0.71 | 0.883±0.003 | 0.765±0.004 | 51.89±0.43 | **67.53±0.47** | 76.91±0.29 | 82.40±0.38 |
| SEABED | **0.675±0.007** | **47.40±0.38** | **0.885±0.002** | **0.769±0.003** | **52.88±0.38** | 67.32±0.49 | **77.27±0.50** | **82.52±0.40** |
| **LUBM** | GED | | Ranking | | | | | |
| | MAE | Accuracy(%) | $\gamma$ | $\tau$ | $p$@5(%) | $p$@10(%) | $p$@15(%) | $p$@20(%) |
| w/o Local | 0.615±0.013 | 51.17±0.90 | 0.917±0.003 | 0.812±0.004 | 65.29±0.97 | 73.69±0.48 | 79.94±0.66 | 84.65±0.26 |
| w/o Motif-Count | 0.623±0.040 | 50.51±2.40 | 0.917±0.005 | 0.813±0.008 | 65.47±1.35 | 73.77±0.71 | 80.18±0.92 | 84.82±0.57 |
| w/o EMD | 0.603±0.015 | 51.89±1.25 | 0.919±0.005 | 0.816±0.007 | **66.75±1.28** | 73.80±0.98 | 80.13±0.79 | 84.73±0.39 |
| w/o Pos.-Emb. | 0.608±0.007 | 50.85±0.69 | 0.918±0.003 | 0.814±0.004 | 66.32±0.35 | 73.73±0.85 | 80.25±0.36 | 84.61±0.34 |
| SEABED | **0.590±0.011** | **52.29±1.05** | **0.924±0.002** | **0.824±0.003** | 66.73±1.14 | **74.48±0.57** | **80.72±0.47** | **85.23±0.23** |
| **YAGO** | GED | | Ranking | | | | | |
| | MAE | Accuracy(%) | $\gamma$ | $\tau$ | $p$@5(%) | $p$@10(%) | $p$@15(%) | $p$@20(%) |
| w/o Local | 0.218±0.067 | 91.76±7.64 | 0.991±0.001 | 0.951±0.001 | 87.88±0.33 | 93.43±0.25 | 95.78±0.21 | 97.03±0.07 |
| w/o Motif-Count | 0.222±0.016 | 91.37±2.76 | 0.991±0.000 | 0.951±0.000 | 87.52±0.26 | 93.46±0.25 | 95.72±0.20 | 97.03±0.12 |
| w/o EMD | 0.215±0.062 | 92.88±7.86 | 0.991±0.000 | **0.952±0.000** | 86.98±0.42 | 93.37±0.14 | 95.81±0.18 | **97.12±0.08** |
| w/o Pos.-Emb. | 0.217±0.027 | 92.43±2.32 | 0.991±0.000 | 0.951±0.000 | 87.50±0.58 | 93.49±0.21 | **95.92±0.16** | 97.02±0.06 |
| SEABED | **0.177±0.032** | **95.78±2.38** | **0.991±0.000** | 0.951±0.000 | **87.88±0.33** | **93.68±0.12** | 95.70±0.08 | 97.06±0.07 |
| **WIKIDATA** | GED | | Ranking | | | | | |
| | MAE | Accuracy(%) | $\gamma$ | $\tau$ | $p$@5(%) | $p$@10(%) | $p$@15(%) | $p$@20(%) |
| w/o Local | 0.284±0.037 | 84.87±4.67 | 0.991±0.000 | 0.948±0.001 | 73.31±0.45 | 90.69±0.32 | 92.80±0.27 | 95.45±0.15 |
| w/o Motif-Count | 0.268±0.024 | 86.48±2.61 | 0.992±0.000 | 0.949±0.001 | 72.99±0.55 | 90.42±0.26 | 92.54±0.31 | 95.55±0.09 |
| w/o EMD | 0.288±0.088 | 83.86±11.52 | 0.992±0.000 | 0.948±0.002 | 73.91±0.76 | 90.51±0.28 | 92.64±0.32 | **95.60±0.16** |
| w/o Pos.-Emb. | 0.274±0.027 | 86.04±3.21 | 0.992±0.000 | 0.948±0.001 | 73.25±0.61 | 90.49±0.32 | **92.82±0.21** | 95.41±0.21 |
| SEABED | **0.242±0.030** | **89.48±1.95** | **0.992±0.000** | **0.949±0.001** | **73.91±0.76** | **90.90±1.10** | 92.60±0.04 | 95.48±0.16 |

4.52%. (3) EMD, as an auxiliary loss function, increases the model's focus on global alignment, but the improvement is limited, with an average MAE reduction of 5.70% and an accuracy improvement of 1.1%. (4) The positional encoding effectively captures the spatial dependencies among sub-KGs and their contextual roles within the entire KG. Removing it leads to a performance drop, for example, on the YAGO dataset, removing positional encoding increases the average MAE from 0.177 to 0.217.

**2.Sensitivity to embedding models.** In this experiment, we include and test four embedding methods for obtaining the initial KG embeddings. Specifically, we include four representative approaches: (1) a GNN-based method, CompGCN [70], (2) a semantic matching-based method, DistMult [82], (3) a translation distance-based method, TransR [42], and (4) the RDF2Vec [64]. For the SWDF and LUBM datasets, we replaced the original embedding method RDF2Vec in SEABED with each of the three embedding models discussed above. Accordingly, SEABED (CompGCN), SEABED (DistMult), SEABED (TransR), and SEABED (RDF2Vec) denote the variants that use CompGCN, DistMult, TransR, and RDF2Vec as their initial embedding models, respectively. The corresponding results are reported in Table 5. As shown in the results, the performance of SEABED remains consistently strong across different embedding choices, demonstrating that its improvements do not rely on a specific embedding model. Besides, under all embedding settings SEABED still outperforms the strongest baseline GEDIOT, further confirming that the effectiveness of our method stems from its semantics-aware design rather than the initial embeddings.

**3. Memory requirements.** The design of SEABED incurs slightly higher GPU usage compared to existing methods. We note that SEABED requires about 2× the GPU memory of baseline models. The detailed result is shown in our technical report [88]. However, this overhead is very modest in practice; the total memory consumption remains under 3GB, which is easily supported by mainstream GPUs (e.g., RTX 3090/4090) as well as common personal GPUs (e.g., RTX 4060 Ti 8GB). Given the significant performance gains achieved by SEABED, we believe this small GPU overhead is unlikely to pose a practical limitation in real-world applications.

**4. Scalability study.** Based on the SWDF dataset, we constructed four expanded versions by progressively increasing the sizes of the KG pairs in each dataset. Here, SWDF-1 corresponds to the original dataset, while higher indices indicate proportionally larger KGs (with SWDF-5 being the largest). The statistics of the five datasets are shown in our technical report [88]. Next, we evaluate SEABED and compare it with the strongest baseline GEDIOT across all five datasets. As shown in Table 6, SEABED consistently outperforms GEDIOT in both MAE and Accuracy across all scales, demonstrating the strong scalability of our method.

**5. Failure cases analysis.** When two KGs exhibit strong global similarity, the global signal may dominate the final prediction, preventing the local semantic differences from fully influencing the estimated GED and causing SEABED to underestimate the true distance. We present the detailed analysis in our technical report [88].

**Table 5: Comparative performance results of SEABED and its variants under different embedding models.**

| SWDF | GED | | Ranking | | | | | |
|---|---|---|---|---|---|---|---|---|
| | MAE | Accuracy(%) | $\gamma$ | $\tau$ | $p$@5(%) | $p$@10(%) | $p$@15(%) | $p$@20(%) |
| SEABED (TransR) | 0.640±0.024 | 48.95±3.14 | 0.894±0.003 | 0.782±0.004 | 54.97±0.53 | 69.51±0.59 | 78.30±0.35 | 83.41±0.34 |
| SEABED (CompGCN) | 0.645±0.008 | 49.06±0.99 | 0.895±0.003 | 0.782±0.004 | 55.67±1.17 | 69.33±1.01 | 78.35±0.62 | 83.04±0.35 |
| SEABED (DistMult) | 0.679±0.010 | 47.00±0.76 | 0.883±0.002 | 0.765±0.003 | 52.37±1.03 | 67.15±1.12 | 76.67±0.27 | 82.06±0.31 |
| SEABED (RDF2Vec) | 0.675±0.007 | 47.40±0.38 | 0.885±0.002 | 0.769±0.003 | 52.88±0.38 | 67.32±0.49 | 77.27±0.50 | 82.52±0.40 |
| **LUBM** | GED | | Ranking | | | | | |
| | MAE | Accuracy(%) | $\gamma$ | $\tau$ | $p$@5(%) | $p$@10(%) | $p$@15(%) | $p$@20(%) |
| SEABED (TransR) | 0.573±0.021 | 53.66±1.83 | 0.926±0.005 | 0.826±0.007 | 67.11±1.23 | 75.30±0.99 | 80.96±0.95 | 85.44±0.69 |
| SEABED (CompGCN) | 0.592±0.010 | 52.06±1.31 | 0.924±0.002 | 0.824±0.003 | 66.95±0.73 | 74.93±0.56 | 81.29±0.35 | 85.31±0.41 |
| SEABED (DistMult) | 0.597±0.010 | 52.22±0.68 | 0.922±0.002 | 0.820±0.003 | 66.41±1.02 | 74.05±0.65 | 80.58±0.13 | 87.03±4.11 |
| SEABED (RDF2Vec) | 0.590±0.011 | 52.29±1.05 | 0.924±0.002 | 0.824±0.003 | 66.73±1.14 | 74.48±0.57 | 80.72±0.47 | 85.23±0.23 |

**Table 6: Comparative results of SEABED and GEDIOT on different scale datasets.**

| SWDF-1 | MAE | Accuracy(%) |
|---|---|---|
| GEDIOT | 0.771±0.068 | 40.57±2.29 |
| SEABED | **0.675±0.007** | **47.40±0.38** |
| **SWDF-2** | MAE | Accuracy(%) |
| GEDIOT | 0.973±0.139 | 30.51±3.71 |
| SEABED | **0.569±0.031** | **50.51±2.83** |
| **SWDF-3** | MAE | Accuracy(%) |
| GEDIOT | 2.289±0.068 | 12.91±0.34 |
| SEABED | **0.579±0.009** | **49.01±0.18** |
| **SWDF-4** | MAE | Accuracy(%) |
| GEDIOT | 2.444±0.172 | 11.65±0.31 |
| SEABED | **1.780±0.741** | **21.19±11.25** |
| **SWDF-5** | MAE | Accuracy(%) |
| GEDIOT | 2.863±0.224 | 10.06±0.57 |
| SEABED | **2.078±0.676** | **17.46±9.55** |

## 6 RELATED WORKS

This section reviews the existing works of graph similarity problem and subgraph matching problem.

• **Graph similarity problem.** Computing similarity between two graphs is a fundamental problem in the graph analysis area, where graph edit distance (GED) [4, 5, 17, 29, 36, 54, 56, 57, 83] and maximum common subgraph (MCS) [6, 43, 45, 58, 74] are the widely used metrics. GED denotes the smallest operations needed to transfer one graph into another, which is widely used in the graph database and graph queries. The GED can be computed via minimum weight matching [17, 29, 36, 56, 57] and A* [49] searches. In general, those traditional solutions are suitable for small graphs, but their performance declines for larger graphs. The minimum weight matching-based methods are based on the key fact that two graphs can be transformed into a bipartite graph. Hence, the GED over these two graphs is exactly the minimum weight matching of this bipartite partial graph. Consequently, researchers have turned to machine learning-based algorithms [4, 5, 12, 54, 83] to enhance efficiency and accuracy. The representative methods are extensively reviewed in Section 3. MCS is the largest subgraph that is commonly present in both input graphs [43, 45, 58, 74]. MCS detection is a well-known NP-hard problem [58] with existing methods based on constraint programming [45, 74], branch and bound [43, 46], integer programming [3], conversion to maximum clique detection [40, 45] and machine-learning based methods [6, 43].

• **Subgraph matching problem.** Due to the importance of subgraph matching, various algorithms [8, 24, 30, 33, 67, 69, 73, 85] have been proposed. The solutions of subgraph matching can be divided into three categories, namely join-based, exploration-based, and hybrid methods. To enable parallel computing, join-based methods have been widely adopted in parallel [34, 59] or distributed manner [31, 37, 38, 55, 77]. For join-based methods, Lai et al. [39] classify join strategies into binary join [37, 38, 55, 68] and worst-case optimal join (WCOJ) [1, 2, 31, 50–52, 72]. Most in-memory single-threaded approaches [8, 24, 25, 30, 33, 73] adopt exploration-based methods. Wang et al. [77] develop an exploration-based approach in distributed systems while researchers [84] explore hybrid methodologies. Subgraph matching can be used in the subgraph search problem [6], which aims to find all the graphs in the database that contain the query graph.

## 7 CONCLUSIONS

In this paper, we investigate the problem of graph edit distance (GED) computation over a pair of KGs. Traditional GED methods are time-consuming and often yield suboptimal results. GNN-based approaches, while effective for simple graphs, typically ignore the rich semantics and complex relationships inherent in KGs, making them unsuitable for direct application to KGs. To improve estimation accuracy, we propose SEABED, which captures both structural and semantic differences between two KGs from local and global perspectives. This is achieved through a novel local semantic alignment module and a new global semantic alignment strategy. Extensive experiments on four real-world KGs demonstrate that our proposed algorithm outperforms the state-of-the-art methods on all datasets. In particular, the mean absolute error is reduced by up to 66.7%, while the accuracy is improved by up to 70.5%. In future work, we plan to leverage authentic KG version histories (e.g., version identifiers, temporal snapshots, or change logs) to construct datasets whose ground-truth GEDs more accurately reflect real-world KG updates, and test the performance of our method on such datasets.

## 8 ACKNOWLEDGMENTS

This work was partially supported by the NSFC under Grant 62302421, the Basic and Applied Basic Research Fund of Guangdong Province under Grant 2025A1515010439, the Guangdong Provincial Key Laboratory of Big Data Computing at The Chinese University of Hong Kong, Shenzhen, and the 1+1+1 CUHK–CUHK(SZ)–GDSTC Joint Collaboration Fund under Grant 2025A0505000045.

# REFERENCES


[1] Khaled Ammar, Frank McSherry, Semih Salihoglu, and Manas Joglekar. 2018. Distributed evaluation of subgraph queries using worst-case optimal low-memory dataflows. *Proc. VLDB Endow.* 11, 6 (Feb. 2018), 691–704. https://doi.org/10.14778/3184470.3184473

[2] Molham Aref, Balder ten Cate, Todd J. Green, Benny Kimelfeld, Dan Olteanu, Emir Pasalic, Todd L. Veldhuizen, and Geoffrey Washburn. 2015. Design and Implementation of the LogicBlox System. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (Melbourne, Victoria, Australia) *(SIGMOD '15)*. Association for Computing Machinery, New York, NY, USA, 1371–1382. https://doi.org/10.1145/2723372.2742796

[3] Laura Bahiense, Gordana Manić, Breno Piva, and Cid C De Souza. 2012. The maximum common edge subgraph problem: A polyhedral investigation. *Discrete Applied Mathematics* 160, 18 (2012), 2523–2541.

[4] Jiyang Bai and Peixiang Zhao. 2021. TaGSim: type-aware graph similarity learning and computation. *Proc. VLDB Endow.* 15, 2 (oct 2021), 335–347. https://doi.org/10.14778/3489496.3489513

[5] 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.

[6] Yunsheng Bai, Derek Xu, Yizhou Sun, and Wei Wang. 2021. Glsearch: Maximum common subgraph detection via learning to search. In *International Conference on Machine Learning*. PMLR, 588–598.

[7] Fei Bi, Lijun Chang, Xuemin Lin, Lu Qin, and Wenjie Zhang. 2016. Efficient subgraph matching by postponing cartesian products. In *Proceedings of the 2016 International Conference on Management of Data*. 1199–1214.

[8] Fei Bi, Lijun Chang, Xuemin Lin, Lu Qin, and Wenjie Zhang. 2016. Efficient Subgraph Matching by Postponing Cartesian Products. In *Proceedings of the 2016 International Conference on Management of Data* (San Francisco, California, USA) *(SIGMOD '16)*. Association for Computing Machinery, New York, NY, USA, 1199–1214. https://doi.org/10.1145/2882903.2915236

[9] Nicolas Boria, Sébastien Bougleux, and Luc Brun. 2018. Approximating GED using a stochastic generator and multistart IPFP. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 460–469.

[10] Horst Bunke. 1997. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognit. Lett.* 18 (1997), 689–694. https://api.semanticscholar.org/CorpusID:21706450

[11] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 3438–3445.

[12] Qihao Cheng, Da Yan, Tianhao Wu, Zhongyi Huang, and Qin Zhang. 2025. Computing Approximate Graph Edit Distance via Optimal Transport. *Proc. ACM Manag. Data* 3, 1, Article 23 (Feb. 2025), 26 pages. https://doi.org/10.1145/3709673

[13] Marco Cuturi. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *NIPS*. 2292–2300.

[14] Richard Cyganiak, David Hyland-Wood, and Markus Lanthaler. 2014. RDF 1.1 Concepts and Abstract Syntax. *W3C Proposed Recommendation* (01 2014).

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=YicbFdNTTy

[16] Christopher L Ebsch, Joseph A Cottam, Natalie C Heller, Rahul D Deshmukh, and George Chin. 2020. Using graph edit distance for noisy subgraph matching of semantic property graphs. In *2020 IEEE international conference on big data (big data)*. IEEE, 2520–2525.

[17] Stefan Fankhauser, Kaspar Riesen, and Horst Bunke. 2011. Speeding Up Graph Edit Distance Computation through Fast Bipartite Matching. In *GbRPR (Lecture Notes in Computer Science, Vol. 6658)*. Springer, 102–111.

[18] Andreas Fischer, Ching Y Suen, Volkmar Frinken, Kaspar Riesen, and Horst Bunke. 2013. A fast matching algorithm for graph-based handwriting recognition. In *Graph-Based Representations in Pattern Recognition: 9th IAPR-TC-15 International Workshop, GbRPR 2013, Vienna, Austria, May 15-17, 2013. Proceedings 9*. Springer, 194–203.

[19] 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.

[20] Carlos Garcia-Hernandez, Alberto Fernández, and Francesc Serratosa. 2020. Learning the edit costs of graph edit distance applied to ligand-based virtual screening. *Current topics in medicinal chemistry* 20, 18 (2020), 1582–1592.

[21] Karam Gouda and Mosab Hassaan. 2016. CSI GED: An Efficient Approach for Graph Edit Similarity Computation. https://doi.org/10.1109/ICDE.2016.7498246

[22] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. 2005. LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.* 3, 2-3 (2005), 158–182. http://dblp.uni-trier.de/db/journals/ws/ws3.html#GuoPH05

[23] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*. 1024–1034.

[24] Myoungji Han, Hyunjoon Kim, Geonmo Gu, Kunsoo Park, and Wook-Shin Han. 2019. Efficient Subgraph Matching: Harmonizing Dynamic Programming, Adaptive Matching Order, and Failing Set Together. In *Proceedings of the 2019 International Conference on Management of Data* (Amsterdam, Netherlands) *(SIGMOD '19)*. Association for Computing Machinery, New York, NY, USA, 1429–1446. https://doi.org/10.1145/3299869.3319880

[25] Huahai He and Ambuj K. Singh. 2008. Graphs-at-a-time: query language and access methods for graph databases. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (Vancouver, Canada) *(SIGMOD '08)*. Association for Computing Machinery, New York, NY, USA, 405–418. https://doi.org/10.1145/1376616.1376660

[26] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge Graphs. *Comput. Surveys* 54, 4 (July 2021), 1–37. https://doi.org/10.1145/3447772

[27] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. arXiv:1905.12265 [cs.LG] https://arxiv.org/abs/1905.12265

[28] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems* 33, 2 (2021), 494–514.

[29] Roy Jonker and Ton Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* 38 (1987), 325–340. https://api.semanticscholar.org/CorpusID:7806079

[30] Alpár Jüttner and Péter Madarasi. 2018. VF2++—An improved subgraph isomorphism algorithm. *Discrete Applied Mathematics* 242 (03 2018). https://doi.org/10.1016/j.dam.2018.02.018

[31] Chathura Kankanamge, Siddhartha Sahu, Amine Mhedbhi, Jeremy Chen, and Semih Salihoglu. 2017. Graphflow: An Active Graph Database. In *Proceedings of the 2017 ACM International Conference on Management of Data* (Chicago, Illinois, USA) *(SIGMOD '17)*. Association for Computing Machinery, New York, NY, USA, 1695–1698. https://doi.org/10.1145/3035918.3056445

[32] George Karypis and Vipin Kumar. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing* 20, 1 (1998), 359–392. https://doi.org/10.1137/S1064827595287997 arXiv:https://doi.org/10.1137/S1064827595287997

[33] Hyunjoon Kim, Yunyoung Choi, Kunsoo Park, Xuemin Lin, Seok-Hee Hong, and Wook-Shin Han. 2021. Versatile Equivalences: Speeding up Subgraph Query Processing and Subgraph Matching. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 925–937. https://doi.org/10.1145/3448016.3457265

[34] Hyeonji Kim, Juneyoung Lee, Sourav S. Bhowmick, Wook-Shin Han, JeongHoon Lee, Seongyun Ko, and Moath H.A. Jarrah. 2016. DUALSIM: Parallel Subgraph Enumeration in a Massive Graph on a Single Machine. In *Proceedings of the 2016 International Conference on Management of Data* (San Francisco, California, USA) *(SIGMOD '16)*. Association for Computing Machinery, New York, NY, USA, 1231–1245. https://doi.org/10.1145/2882903.2915209

[35] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR (Poster)*. OpenReview.net.

[36] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[37] Longbin Lai, Lu Qin, Xuemin Lin, and Lijun Chang. 2015. Scalable subgraph enumeration in MapReduce. *Proc. VLDB Endow.* 8, 10 (June 2015), 974–985. https://doi.org/10.14778/2794367.2794368

[38] Longbin Lai, Lu Qin, Xuemin Lin, Ying Zhang, Lijun Chang, and Shiyu Yang. 2016. Scalable distributed subgraph enumeration. *Proc. VLDB Endow.* 10, 3 (Nov. 2016), 217–228. https://doi.org/10.14778/3021924.3021937

[39] Longbin Lai, Zhu Qing, Zhengyi Yang, Xin Jin, Zhengmin Lai, Ran Wang, Kongzhang Hao, Xuemin Lin, Lu Qin, Wenjie Zhang, Ying Zhang, Zhengping Qian, and Jingren Zhou. 2019. Distributed subgraph matching on timely dataflow. *Proc. VLDB Endow.* 12, 10 (June 2019), 1099–1112. https://doi.org/10.14778/3339490.3339494

[40] Giorgio Levi. 1973. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo* 9, 4 (1973), 341–352.

[41] Yongjiang Liang and Peixiang Zhao. 2017. Similarity Search in Graph Databases: A Multi-Layered Indexing Approach. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. 783–794. https://doi.org/10.1109/ICDE.2017.129

[42] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 29.

[43] Yanli Liu, Chu-Min Li, Hua Jiang, and Kun He. 2020. A Learning Based Branch and Bound for Maximum Common Subgraph Related Problems. In *AAAI*. AAAI Press, 2392–2399.

[44] Ray M Marin, Nestor F Aguirre, and Edgar E Daza. 2008. Graph theoretical similarity approach to compare molecular electrostatic potentials. *Journal of chemical information and modeling* 48, 1 (2008), 109–118.

[45] Ciaran McCreesh, Samba Ndojh Ndiaye, Patrick Prosser, and Christine Solnon. 2016. Clique and constraint models for maximum common (connected) subgraph problems. In *International Conference on Principles and Practice of Constraint Programming*. Springer, 350–368.

[46] Ciaran McCreesh, Patrick Prosser, and James Trimble. 2017. A Partitioning Algorithm for Maximum Common Subgraph Problems. In *IJCAI*. ijcai.org, 712–719.

[47] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR (Workshop Poster)*.

[48] Knud Möller, Tom Heath, Siegfried Handschuh, and John Domingue. 2007. Recipes for Semantic Web Dog Food - The ESWC and ISWC Metadata Projects.. In *ISWC/ASWC (Lecture Notes in Computer Science, Vol. 4825)*, Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux (Eds.). Springer, 802–815. http://dblp.uni-trier.de/db/conf/semweb/iswc2007.html#MollerHHD07

[49] Michel Neuhaus, Kaspar Riesen, and Horst Bunke. 2006. Fast Suboptimal Algorithms for the Computation of Graph Edit Distance. In *SSPR/SPR*. https://api.semanticscholar.org/CorpusID:9188232

[50] Hung Q. Ngo. 2018. Worst-Case Optimal Join Algorithms: Techniques, Results, and Open Problems. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (Houston, TX, USA) *(PODS '18)*. Association for Computing Machinery, New York, NY, USA, 111–124. https://doi.org/10.1145/3196959.3196990

[51] Hung Q. Ngo, Ely Porat, Christopher Ré, and Atri Rudra. 2012. Worst-case optimal join algorithms: [extended abstract]. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (Scottsdale, Arizona, USA) *(PODS '12)*. Association for Computing Machinery, New York, NY, USA, 37–48. https://doi.org/10.1145/2213556.2213565

[52] Hung Q Ngo, Christopher Ré, and Atri Rudra. 2014. Skew strikes back: new developments in the theory of join algorithms. *SIGMOD Rec.* 42, 4 (Feb. 2014), 5–16. https://doi.org/10.1145/2590989.2590991

[53] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. 2019. Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it's done. *Queue* 17, 2 (2019), 48–75.

[54] Chengzhi Piao, Tingyang Xu, Xianguo Sun, Yu Rong, Kangfei Zhao, and Hong Cheng. 2023. Computing Graph Edit Distance via Neural Graph Matching. *Proceedings of the VLDB Endowment* 16 (06 2023), 1817–1829. https://doi.org/10.14778/3594512.3594514

[55] Miao Qiao, Hao Zhang, and Hong Cheng. 2017. Subgraph matching: on compression and computation. *Proc. VLDB Endow.* 11, 2 (Oct. 2017), 176–188. https://doi.org/10.14778/3149193.3149198

[56] Romain Raveaux, Jean-Christophe Burie, and Jean-Marc Ogier. 2010. A graph matching method and a graph matching distance based on subgraph assignments. *Pattern Recognition Letters* 31 (04 2010), 394–406. https://doi.org/10.1016/j.patrec.2009.10.011

[57] Kaspar Riesen and Horst Bunke. 2009. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing* 27 (06 2009), 950–959. https://doi.org/10.1016/j.imavis.2008.04.004

[58] Kenneth H Rosen. 2007. *Discrete mathematics and its applications*. The McGraw Hill Companies,.

[59] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. 2021. Knowledge Graph Embedding for Link Prediction: A Comparative Analysis. *ACM Trans. Knowl. Discov. Data* 15, 2, Article 14 (jan 2021), 49 pages. https://doi.org/10.1145/3424672

[60] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 1998. *The Earth Mover"s Distance as a Metric for Image Retrieval*. Technical Report. Stanford, CA, USA.

[61] Tim Schwabe and Maribel Acosta. 2024. Cardinality Estimation over Knowledge Graphs with Embeddings and Graph Neural Networks. *Proceedings of the ACM on Management of Data* 2, 1 (March 2024), 1–26. https://doi.org/10.1145/3639299

[62] Ary Mazharudin Shiddiqi, Daniel Siahaan, Hidayatul Munawaroh, et al. 2023. Structural Similarity Assessment of Business Process Graph Using GED-Greedy. In *2023 IEEE 13th International Conference on Control System, Computing and Engineering (ICCSCE)*. IEEE, 45–50.

[63] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. 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/b337e84de8752b27eda3a12363109e80-Paper.pdf

[64] Bram Steenwinckel, Gilles Vandewiele, Terencio Agozzino, and Femke Ongenae. 2023. pyRDF2Vec: A Python Implementation and Extension of RDF2Vec. In *European Semantic Web Conference*. Springer Nature Switzerland, 471–483. https://doi.org/10.1007/978-3-031-33455-9_28

[65] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. YAGO: A Large Ontology from Wikipedia and WordNet. *Web Semant.* 6, 3 (sep 2008), 203–217. https://doi.org/10.1016/j.websem.2008.06.001

[66] Shixuan Sun and Qiong Luo. 2020. In-memory subgraph matching: An in-depth study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1083–1098.

[67] Shixuan Sun, Xibo Sun, Yulin Che, Qiong Luo, and Bingsheng He. 2020. RapidMatch: a holistic approach to subgraph query processing. *Proc. VLDB Endow.* 14, 2 (Oct. 2020), 176–188. https://doi.org/10.14778/3425879.3425888

[68] Zhao Sun, Hongzhi Wang, Haixun Wang, Bin Shao, and Jianzhong Li. 2012. Efficient subgraph matching on billion node graphs. *Proc. VLDB Endow.* 5, 9 (May 2012), 788–799. https://doi.org/10.14778/2311906.2311907

[69] J. R. Ullmann. 1976. An Algorithm for Subgraph Isomorphism. *J. ACM* 23, 1 (Jan. 1976), 31–42. https://doi.org/10.1145/321921.321925

[70] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P Talukdar. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR*.

[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. arXiv:1706.03762 [cs.CL] https://arxiv.org/abs/1706.03762

[72] Todd L. Veldhuizen. 2013. Leapfrog Triejoin: a worst-case optimal join algorithm. arXiv:1210.0481 [cs.DB] https://arxiv.org/abs/1210.0481

[73] Carletti Vincenzo, Pasquale Foggia, Alessia Saggese, and Mario Vento. 2017. Challenging the Time Complexity of Exact Subgraph Isomorphism for Huge and Dense Graphs with VF3. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (04 2017), 1–1. https://doi.org/10.1109/TPAMI.2017.2696940

[74] Philippe Vismara and Benoît Valery. 2008. Finding maximum common connected subgraphs using clique detection or constraint satisfaction algorithms. In *International Conference on Modelling, Computation and Optimization in Information Systems and Management Sciences*. Springer, 358–368.

[75] Xiaoli Wang, Xiaofeng Ding, Anthony K. H. Tung, Shanshan Ying, and Hai Jin. 2012. An Efficient Graph Indexing Method. In *ICDE*. IEEE Computer Society, 210–221.

[76] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2020. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. arXiv:1911.06136 [cs.CL] https://arxiv.org/abs/1911.06136

[77] Zhaokang Wang, Weiwei Hu, Guowang Chen, Chunfeng Yuan, Rong Gu, and Yihua Huang. 2021. Towards Efficient Distributed Subgraph Enumeration Via Backtracking-Based Framework. *IEEE Transactions on Parallel and Distributed Systems* 32, 12 (2021), 2953–2969. https://doi.org/10.1109/TPDS.2021.3076246

[78] Tobias Weller and Maribel Acosta. 2021. Predicting instance type assertions in knowledge graphs using stochastic neural networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2111–2118.

[79] Wikipedia. 2025. $p$-value. https://en.wikipedia.org/wiki/P-value.

[80] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (Jan. 2021), 4–24. https://doi.org/10.1109/tnnls.2020.2978386

[81] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*. OpenReview.net.

[82] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR (Poster)*.

[83] 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.

[84] Yuejia Zhang, Weiguo Zheng, Zhijie Zhang, Peng Peng, and Xuecang Zhang. 2022. Hybrid Subgraph Matching Framework Powered by Sketch Tree for Distributed Systems. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 1031–1043. https://doi.org/10.1109/ICDE53745.2022.00082

[85] Zhijie Zhang, Yujie Lu, Weiguo Zheng, and Xuemin Lin. 2024. A Comprehensive Survey and Experimental Study of Subgraph Matching: Trends, Unbiasedness, and Interaction. *Proceedings of the ACM on Management of Data* 2, 1 (2024), 1–29.

[86] Weiguo Zheng, Lei Zou, Xiang Lian, Dong Wang, and Dongyan Zhao. 2015. Efficient Graph Similarity Search Over Large Graph Databases. *Knowledge and Data Engineering, IEEE Transactions on* 27 (04 2015), 964–978. https://doi.org/10.1109/TKDE.2014.2349924

[87] Yingli Zhou, Yixiang Fang, Chenhao Ma, Tianci Hou, and Xin Huang. 2024. Efficient Maximal Motif-Clique Enumeration over Large Heterogeneous Information Networks. *Proceedings of the VLDB Endowment* 17, 11 (2024), 2946–2959.

[88] Yingli Zhou, Wang Huizhong, Ma Chenhao, and Yixiang Fang. 2025. A Semantics-aware Approach for Graph Edit Distance Estimation over Knowledge Graphs (technical report). https://github.com/JayLZhou/TechnicalReport/blob/main/GEDKG_Technical_Report.pdf.