2018 IEEE 34th International Conference on Data Engineering

An Efficient Probabilistic Approach for Graph Similarity Search

Zijian Li #1, Xun Jian #2, Xiang Lian *3, Lei Chen #4

Computer Science and Engineering Department, HKUST, Hong Kong, China
1 zlicb@cse.ust.hk 2 xjian@cse.ust.hk 4 leichen@cse.ust.hk

* Computer Science Department, Kent State University, Kent, USA
3 xlian@kent.edu

***Abstract*—Graph similarity search is a common and fundamental operation in graph databases. One of the most popular graph similarity measures is the Graph Edit Distance (GED) mainly because of its broad applicability and high interpretability. Despite its prevalence, exact GED computation is proved to be NP-hard, which could result in unsatisfactory computational efficiency on large graphs. However, exactly accurate search results are usually unnecessary for real-world applications especially when the responsiveness is far more important than the accuracy. Thus, in this paper, we propose a novel probabilistic approach to efficiently estimate GED, which is further leveraged for the graph similarity search. Specifically, we first take branches as elementary structures in graphs, and introduce a novel graph similarity measure by comparing branches between graphs, i.e., Graph Branch Distance (GBD), which can be efficiently calculated in polynomial time. Then, we formulate the relationship between GED and GBD by considering branch variations as the result ascribed to graph edit operations, and model this process by probabilistic approaches. By applying our model, the GED between any two graphs can be efficiently estimated by their GBD, and these estimations are finally utilized in the graph similarity search. Extensive experiments show that our approach has better accuracy, efficiency and scalability than other comparable methods in the graph similarity search over real and synthetic data sets.**

I. Introduction

Graph similarity search is a common and fundamental operation in graph databases, which has widespread applications in various fields including bio-informatics, sociology, and chemical analysis, over the past few decades. For evaluating the similarity between graphs, *Graph Edit Distance* (GED) [1] is one of the most prevalent measures because of its wide applicability, that is, GED is capable of dealing with various kinds of graphs including directed and undirected graphs, labeled and unlabeled graphs, as well as simple graphs and multi-graphs (which could have multiple edges between two vertices). Furthermore, GED has high interpretability, since it corresponds to some sequences of concrete graph edit operations (including insertion of vertices and edges, etc.) of minimal lengths, rather than implicit graph embedding utilized in spectral [2] or kernel [3] measures. Example 1 illustrates the basic idea of GED.

Example 1: Assume that we have two graphs G_1 and G_2 as shown in Figure 1. The label sets of graph G_1's vertices and edges are $\{A, B, C\}$ and $\{y, y, z\}$, respectively, and the label sets of graph G_2's vertices and edges are $\{A, B, C\}$ and $\{x, y, z\}$, respectively. The Graph Edit Distance (GED) between G_1 and G_2 is the minimal number of graph edit operations to transform G_1 into G_2. It can be proved that the GED between G_1 and G_2 is 3, which can be achieved by ① deleting the edge between v_1 and v_3 in G_1, and ② inserting an isolated vertex v_4 with label A, and ③ inserting an edge between v_3 and v_4 with label x.

With GED as the graph similarity measure, the *graph similarity search* problem is formally stated as follows.

Problem Statement: **(Graph Similarity Search)** Given a graph database D, a query graph Q, and a similarity threshold $\hat{\tau}$, the graph similarity search is to find a set of graphs $D_0 \subseteq D$, where the graph edit distance (GED) between Q and each graph in D_0 is less than or equal to $\hat{\tau}$.

A straightforward solution to the problem above is to check exact GEDs for all pairs of Q and graphs in database D. However, despite its prevalence, GED is proved to be NP-hard for exact calculations [4], which may lead to unsatisfactory computational efficiency when we conduct a similarity search over large graphs. The most widely-applied approach for computing exact GED is the A^* algorithm [5], which aims to search out the optimal matching between the vertices of two graphs in a heuristic manner. Specifically, given two graphs with n and m vertices, respectively, the time complexity of A^* algorithm is $O(n^m)$ in the worst case.

Due to the hardness of computing exact GED (NP-hard) [4], most existing works involving exact GED computation [4] [5] only conducted experiments on graphs with less than 10 vertices. In addition, a recent study [6] indicates that the A^* algorithm is incapable of computing GED between graphs with more than 12 vertices, which can hardly satisfy the demand for searching real-world graphs. For instance, a common requirement in bio-informatics is to search and compare molecular structures of similar proteins [7]. However, the structures of human proteins usually contain hundreds of vertices (i.e., atoms) [8], which obviously makes similarity search beyond the capability of the approaches mentioned above. Another observation is that many real-world applications do not always require an exact similarity value, and an approximate one with some quality guarantee is also acceptable especially in real-time applications where the responsiveness is far more important than the accuracy. Taking the protein search as an example again, it is certainly more desirable for users to obtain an approximate solution within a second, rather than to wait for a couple of days to get the exact answer.

To address the problems above, many approaches have been

2375-026X/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDE.2018.00055

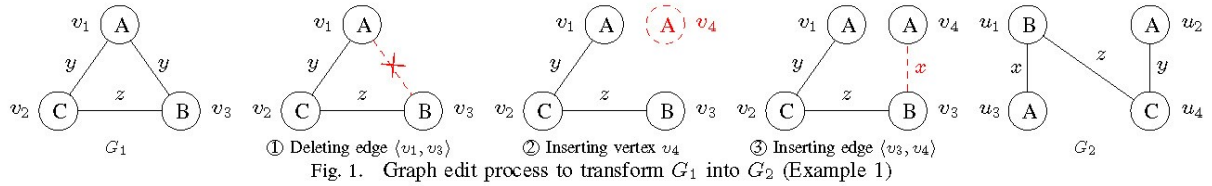

Fig. 1. Graph edit process to transform G_1 into G_2 (Example 1)

proposed to achieve an approximate GED between the query graph Q and each graph in database D in polynomial time [9], which can be leveraged to accelerate the graph similarity search by trading accuracy for efficiency. Assuming that there are two graphs with n and m vertices, respectively, where $n \geq m$, one well-studied method [10] [11] for estimating the GED between these two graphs is to solve a corresponding *linear sum assignment problem*, which requires at least $O(n^3)$ time for obtaining the global optimal value or $O(n^2 \log n^2)$ time for a local optimal value by applying the greedy algorithm [12]. An alternative method is *spectral seriation* [13], which first generates the vector representations of graphs by extracting their leading eigenvalues of the adjacency matrix ($O(n^2)$ time) [14], and then exploits a probabilistic model based on these vectors to estimate GED in $O(nm^2)$ time.

To further enhance the efficiency of GED estimation and better satisfy the demands for graph similarity search on large graphs, we propose a novel probabilistic approach which aims at estimating the GED with less time cost $O(nd+\hat{\tau}^3)$, where n is the number of vertices, d is the average degree of the graphs involved, and $\hat{\tau}$ is the similarity threshold in the stated graph similarity search problem. Note that the similarity threshold $\hat{\tau}$ is often set as a small value (i.e., $\hat{\tau} \leq 10$) and does not increase with the number of vertices n in previous studies [4] [15], thus, we can assume that $\hat{\tau}$ is a constant with regard to n when the graph is sufficiently large. Moreover, most real-world graphs studied in related works [11] [12] are *scale-free graphs* [16], and we prove that the average degree $d = O(\log n)$ for scale-free graphs. Therefore, under the assumptions above, the time complexity of our approach is $O(nd+\hat{\tau}^3) = O(n \log n)$ for comparing two scale-free graphs, and $O(|D| n \log n)$ for searching similar graphs in the graph database D, where $|D|$ is the number of graphs in database D.

Our method is mainly inspired by probabilistic modeling approaches which are broadly utilized in similarity searches on various types of data such as text and images [17]. The basic idea of these methods is to model the formation of an object as a stochastic process, and to establish probabilistic connections between objects. In this paper, we follow this idea and model the formation of graph branch distances (GBDs) as the results of random graph editing processes, where GBD is defined in Section III. By doing so, we prove the probabilistic relationship between GED and GBD, which is finally utilized to estimate GED by graph branch distance (GBD).

To summarize, we make the following contributions.

- We adopt *branches* [15] as elementary structures in graphs, and define a novel graph similarity measure by comparing branches between graphs, i.e., *Graph Branch Distance* (GBD), which can be efficiently calculated in $O(nd)$ time, where n is the number of vertices, and d is the average degree of the graphs involved.
- We build a probabilistic model which reveals the relationship between GED and GBD by considering branch variations as the result ascribed to graph edit operations. By applying our model, the GED between any two graphs can be estimated by their GBD in $O(\hat{\tau}^3)$ time, where $\hat{\tau}$ is the similarity threshold in the graph similarity search problem.
- We conduct extensive experiments to show our approach has better accuracy, efficiency and scalability compared with the related approximation methods over real and synthetic data.

The paper is organized as follows. In Section II, we formally define the symbols and concepts which are used in this paper. In Section III, we give definitions of branches and *Graph Branch Distance* (GBD). In Section IV, we introduce the *extended graphs*, which are exploited to simplify our model. In Section V, we derive the probabilistic relation between GBD and GED, which is leveraged in Section VI to perform the graph similarity search. In Section VII, we demonstrate the efficiency and effectiveness of our proposed approaches through extensive experiments. We discuss related works in Section VIII, and we conclude the paper in Section IX.

II. Preliminaries

The *graphs* discussed in this paper are restricted to *simple labeled undirected graphs*. Specifically, the i-th graph in database D is denoted by: $G_i \triangleq \{V_i, E_i, \mathcal{L}\}$, where $V_i \triangleq \{v_{i,1}, v_{i,2}, ..., v_{i,|V_i|}\}$ is the set of vertices, $E_i \triangleq \{e_{i,1}, e_{i,2}, ..., e_{i,|E_i|}\}$ is the set of edges, while $\mathcal{L}$ is a general labelling function. For any vertex $v_{i,j} \in V_i$, its label is given by $\mathcal{L}(v_{i,j})$. Similarly, for any edge $e_{i,j} \in E_i$, its label is given by $\mathcal{L}(e_{i,j})$. In addition, $\mathcal{L}_V$ and $\mathcal{L}_E$ are defined as the sets of all possible labels for vertices and edges, respectively. We also define ε as a *virtual* label, which will be used later in our approach. When the label of a vertex (or edge) is ε, the vertex (or edge) is said to be *virtual* and does not actually exist. Particularly, we have $\varepsilon \notin \mathcal{L}_V$ and $\varepsilon \notin \mathcal{L}_E$. Note that our method can also handle directed and weighted graphs by considering edge directions and weights as special labels.

In this paper, we take Graph Edit Distance (GED) [1] as the graph similarity measure, which is defined as follows.

Definition 1 (Graph Edit Distance): The edit distance between graphs G_1 and G_2, denoted by $GED(G_1, G_2)$, is the minimal number of graph edit operations which are necessary to transform G_1 into G_2, where the graph edit operations (GEO) are restricted to the following six types:

- **AV**: Add one isolated vertex with a non-virtual label;
- **DV**: Delete one isolated vertex;
- **RV**: Relabel one vertex;
- **AE**: Add one edge with a non-virtual label;
- **DE**: Delete one edge;
- **RE**: Relabel one edge.

Assume that a particular graph edit operation sequence which transforms graph G_1 into G_2 is seq_i, where i is the

unique ID of this sequence. Then, according to Definition 1, $GED(G_1, G_2)$ is the minimal length for all possible operation sequences, that is, $GED(G_1, G_2) = \min_i\{|seq_i|\}$, where $|seq_i|$ is the length of the sequence seq_i. The set of all operation sequences from G_1 to G_2 of the minimal length is defined as $SEQ = \{seq_i | \forall i, GED(G_1, G_2) = |seq_i|\}$.

III. Branch Distance Between Graphs

To reduce the high cost of exact GED computations (NP-hard [4]) in the graph similarity search, one widely-applied strategy for pruning search results [4] [18] [19] [15] is to exploit the differences between graph sub-structures as the bounds of exact GED values. In this paper, we consider the branches [15] as elementary graph units, which are defined as:

Definition 2 (Branches): The branch rooted at vertex v is defined as $B(v) = \{\mathcal{L}(v), N(v)\}$, where $\mathcal{L}(v)$ is the label of vertex v, and $N(v)$ is the sorted multiset containing all labels of edges adjacent to v. The sorted multiset of all branches in G_i is denoted by $B_{G_i} = \{B(v_{i,j}), \forall v_{i,j} \in V_i\}$.

In practice, each branch $B(v)$ is stored as a list of strings whose first element is $\mathcal{L}(v)$ and the following elements are strings in the sorted multiset $N(v)$. In addition, B_{G_i} for each graph G_i is stored in a sorted multiset, whose elements are essentially lists of strings (i.e., branches) and are always sorted ascendingly by the ordering algorithm in [20]. For a fair comparison of the computational efficiency, we assume that all auxiliary data structures in different methods, such as the cost matrix [11], adjacency matrix [13], and our branches, are pre-computed and stored with graphs.

To define the equality between two branches, we introduce the concept of *branch isomorphism* as follows.

Definition 3 (Branch Isomorphism): Two branches $B(v) = \{\mathcal{L}(v), N(v)\}$ and $B(u) = \{\mathcal{L}(u), N(u)\}$ are isomorphic if and only if $\mathcal{L}(v) = \mathcal{L}(u)$ and $N(v) = N(u)$, which is denoted by $B(v) \simeq B(u)$.

Suppose that we have two branches $B(v)$ and $B(u)$ where the degrees of vertices v and u are d_v and d_u, respectively. From previous discussions, the branch $B(v)$ and $B(u)$ are stored as lists of lengths $d_v + 1$ and $d_u + 1$, respectively. Therefore, checking whether $B(v)$ and $B(u)$ are isomorphic is essentially judging whether two lists of lengths $d_v + 1$ and $d_u + 1$ are identical, which can be done in d_v time when $d_v = d_u$, and otherwise in one unit time since the length of a list can be obtained in one unit time.

Finally, we define the Graph Branch Distance (GBD).

Definition 4 (Graph Branch Distance): The branch distance between graphs G_1 and G_2, denoted by $GBD(G_1, G_2)$, is defined as:

$$\begin{aligned} GBD(G_1, G_2) &= \max\{|B_{G_1}|, |B_{G_2}|\} - |B_{G_1} \cap B_{G_2}| \\ &= \max\{|V_1|, |V_2|\} - |B_{G_1} \cap B_{G_2}| \end{aligned} \quad (1)$$

where B_{G_1} and B_{G_2} are the multisets of all branches in graphs G_1 and G_2, respectively.

The intuition of introducing GBD is as follows. The state-of-the-art method [15] for pruning graph similarity search results assumes that the difference between branches of two graphs has a close relation to their GED. Therefore, in this paper, we aim to use GBD to closely estimate the GED of two graphs.

TABLE I
Table of Notations

D	$\triangleq$	$\{G_1, G_2, ...G_{\lvert D\rvert}\}$, the graph database
G_i	$\triangleq$	$\{V_i, E_i, \mathcal{L}\}$, i-th graph in database
V_i	$\triangleq$	$\{v_{i,1}, v_{i,2}, ..., v_{i,\lvert V_i\rvert}\}$, the vertices in G_i
E_i	$\triangleq$	$\{e_{i,1}, e_{i,2}, ..., e_{i,\lvert E_i\rvert}\}$, the edges in G_i
Q	$\triangleq$	$\{V_Q, E_Q, \mathcal{L}\}$, the query graph
$\mathcal{L}$	$\triangleq$	labelling function for vertices and edges
$\mathcal{L}_V$	$\triangleq$	the set of all possible vertex labels
$\mathcal{L}_E$	$\triangleq$	the set of all possible edge labels
ε	$\triangleq$	the *virtual* label
$\hat{\tau}$	$\triangleq$	the similarity threshold
seq	$\triangleq$	a graph edit operation (GEO) sequence
SEQ	$\triangleq$	set of all GEO sequences of minimal lengths
GED	$\triangleq$	Graph Edit Distance
$B(v)$	$\triangleq$	the branch rooted at vertex v
$\mathcal{L}(v)$	$\triangleq$	the label of vertex v
$N(v)$	$\triangleq$	sorted multiset of labels of edges adjacent to v
B_{G_i}	$\triangleq$	sorted multiset of all branches in graph G_i
GBD	$\triangleq$	Graph Branch Distance
$G^{\{k\}}$	$\triangleq$	extended graph of G with extension factor k
G'_1, G'_2	$\triangleq$	$G_1^{\{\lvert V_2\rvert - \lvert V_1\rvert\}}$, $G_2^{\{0\}}$ when $\lvert V_1\rvert \leq \lvert V_2\rvert$

Example 2 below illustrates the process of computing GBD.

Example 2: Assume that we have two graphs G_1 and G_2 as shown in Figure 1. The branches rooted at the vertices in graphs G_1 and G_2 are as follows.

$B(v_1) = \{A; y, y\}, B(v_2) = \{C; y, z\}, B(v_3) = \{B; y, z\};$

$B(u_1) = \{B; x, z\}, B(u_2) = \{A; y\};$

$B(u_3) = \{A; x\}, B(u_4) = \{C; y, z\}.$

The sorted multisets of branches in G_1 and G_2 are:

$$B_{G_1} = \{B(v_1), B(v_3), B(v_2)\};$$
$$B_{G_2} = \{B(u_3), B(u_2), B(u_1), B(u_4)\}.$$

Therefore, according to Definition 4, we can obtain the graph branch distance (GBD) between graphs G_1 and G_2 by applying the Equation (1), which is:

$$GBD(G_1, G_2) = \max\{|B_{G_1}|, |B_{G_2}|\} - |B_{G_1} \cap B_{G_2}| = 3,$$

where $|B_{G_1}| = 3$ and $|B_{G_2}| = 4$ are the sizes of multisets B_{G_1} and B_{G_2}, respectively. In addition, according to Definition 3, the only pair of isomorphic branches between multisets B_{G_1} and B_{G_2} is $B(v_2) \simeq B(u_4)$. Therefore, the intersection set of multisets B_{G_1} and B_{G_2} is $\{B(v_2)\}$, whose size is $|B_{G_1} \cap B_{G_2}| = |\{B(v_2)\}| = 1$.

Note that the time cost of computing the *size* of intersection of two *sorted* multisets is $\max\{m_1, m_2\}$ [4], where m_1 and m_2 are the sizes of two multisets, respectively. Therefore, the GBD between query graph Q and any graph $G \in D$ can be computed in time:

$$\textstyle\sum_i^n d_i = O(nd), \quad (2)$$

where $n = \max\{|V_Q|, |V_G|\}$, d_i is the degree of i-th compared vertex in G, and d is the average degree of graph G.

The GBD defined in this section is utilized to model the graph edit process and further leveraged for estimating the graph edit distance (GED) in Section V.

IV. Extended Graphs

In this section, we reduce the number of graph edit operation types that need to be considered by extending graphs with *virtual* vertices and edges, which helps to simplify our probabilistic model in Section V. Moreover, we show that the GED

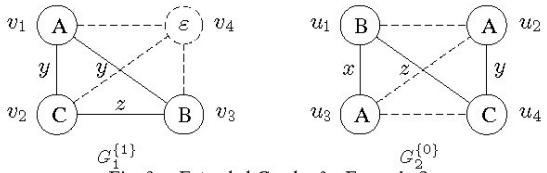

Fig. 2. Extended Graphs for Example 3

and the GBD between the extended graphs stay the same as the GED and the GBD between the original ones, respectively.

The definition of extended graphs is as follows.

Definition 5 (Extended Graphs): For graph G, its extended graph, denoted by $G^{\{k\}}$, is generated by first inserting k isolated virtual vertices into G, and then inserting a virtual edge between each pair of non-adjacent vertices in G (including virtual vertices), where k is the extension factor.

Example 3: For graphs G_1 and G_2 in Figure 1, their extended graphs $G_1^{\{1\}}$ and $G_2^{\{0\}}$ are shown in Figure 2. The virtual vertices are labelled by ε, while virtual edges are represented by dashed lines. Note that when the extension factor is 0, no virtual vertex will be inserted.

In particular, for any graph pair (G_1, G_2) where $|V_1| \leq |V_2|$, we define $G_1' = G_1^{\{|V_2|-|V_1|\}}$ and $G_2' = G_2^{\{0\}}$ by extending G_1 and G_2 with extension factor $|V_2| - |V_1|$ and 0, respectively. Previous studies [21] [22] have shown that, for any graph edit operation sequence seq which transforms the extended graph G_1' into G_2' and has the minimal length, every operation in seq is equivalent to a relabelling operation on G_1'. Therefore, we only need to consider graph edit operations of types **RV** and **RE** when modeling the graph edit process of transforming the extended graph G_1' into G_2'.

Finally, given the graphs G_1 and G_2 (for $|V_1| \leq |V_2|$), and their extended graphs $G_1'$and G_2', we have the following Theorems 1 and 2, which are utilized in the Section V.

Theorem 1: $GED(G_1, G_2) = GED(G_1', G_2')$

Proof: Please refer to Appendix A. ■

Theorem 2: $GBD(G_1, G_2) = GBD(G_1', G_2')$

Proof: Please refer to Appendix B. ■

Note that the extended graph is only a conceptual model for reducing the number of graph edit operation types that need to be considered. According to Theorems 1 and 2, whenever the values of $GED(G_1', G_2')$ and $GBD(G_1', G_2')$ are required, we can instead calculate $GED(G_1, G_2)$ and $GBD(G_1, G_2)$. Therefore, in practice, we do not actually convert graphs into their extended versions, and the calculations of GEDs and GBDs are still conducted on original graphs rather than the extended ones, which means that there is no overhead for creating and maintaining extended graphs.

V. Our Probabilistic Model

In this section, we aim to solve the stated graph similarity search problem by estimating GED from GBD in a probabilistic manner. According to Theorems 1 and 2, the relation between original graphs' GED and GBD must be the same with the relation between extended graphs' GED and GBD. Therefore, as discussed in Section IV, we only need to consider graph edit operations of types **RV** and **RE** when building our probabilistic model.

To be more specific, we consider two given graphs G_1 and G_2 where $|V_1| \leq |V_2|$ and their extended graphs G_1' and G_2'.

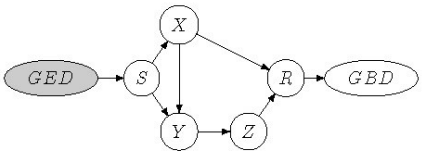

Fig. 3. Bayesian Network of Random Variables

TABLE II
Notation of Random Variables

S	$\triangleq$	Choice of operation sequence
X	$\triangleq$	Number of relabeled vertices
Y	$\triangleq$	Number of relabeled edges
Z	$\triangleq$	Number of vertices covered by relabeled edges
R	$\triangleq$	Number of vertices either relabeled or covered by relabeled edges

For simplicity, we denote $GED(G_1', G_2')$ and $GBD(G_1', G_2')$ by GED and GBD, respectively. As mentioned in Section I, we model the formation of graph branch distances (GBDs) as the results of random graph editing processes, and thus we establish a probabilistic connection between GED and GBD. The detailed steps to construct our model is as follows.

Step 1: We consider GED as an observed random variable.

Step 2: We randomly choose one graph edit operation (GEO) sequence from all possible GEO sequences whose lengths are equal to GED. We define a random variable $X(\omega)$, where ω is a particular choice of operation sequence from SEQ, and $S(\omega) = s$ iff ω choose the sequence with ID s, that is, seq_s.

Step 3: We model the numbers of **RV** and **RE** operations in the sequence chosen in Step 2 as random variables $X(s)$ and $Y(s)$, respectively, where $X(s) = x$ iff the number of operations with type **RV** in seq_s is x, and $Y(s) = y$ iff the number of operations with type **RE** in seq_s is y. Note that, when given $GED = \tau$ and $X = x$, we always have $Y = \tau - x$.

Step 4: We define random variables $Z(y)$ as the random variable where y is a particular value of Y, and $Z(y) = m$ iff $Y(s) = y$ and the number of vertices covered by relabeled edges is m when conducting seq_s. In addition, we define $R(x, m)$ as the random variable where x and m are particular values of X and Z, respectively. That is, $R(x, m) = r$ iff $X = x$, $Z = m$ and the number of vertices either relabelled or covered by relabelled edges is r.

The reason we conduct Step 4 is because we want to model branch variations by the number of branches rooted at the vertices either relabelled or covered by relabelled edges, i.e., the random variable R.

Step 5: We consider GBD as the random variable dependent on R, where their relation is proved in our technical report [23] due to the space limitation.

The random variables defined in the five-step model above are listed in Table II, and their relations among are represented by a Bayesian Network, as shown in Figure 3. We use Example 4 below to better illustrate the key idea of our model.

Example 4: Given two extended graphs G_1' and G_2', as shown in Figure 4, where the virtual edges are represented by dashed lines. The minimal number of graph edit operations to transform G_1' into G_2' is 2, and the set of all possible graph edit operation sequences with the minimal length 2 is:

$$SEQ = \{\{op_1, op_2\}_1, \{op_2, op_1\}_2, \{op_3, op_4\}_3, \{op_4, op_3\}_4\}$$

where the subscript of each sequence is its ID, and

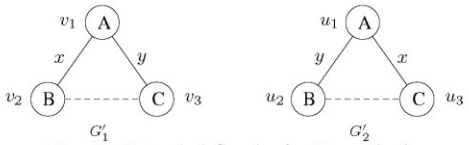

Fig. 4. Extended Graphs for Example 4

$op_1 =$ Relabelling the edge $\langle v_1, v_2 \rangle$ to label y;

$op_2 =$ Relabelling the edge $\langle v_1, v_3 \rangle$ to label x;

$op_3 =$ Relabelling the vertex v_2 to label C.

$op_4 =$ Relabelling the vertex v_3 to label B.

Then, the values of random variables in our model could be:

1) First, we consider GED as a random variable, which has the value 2 in this example. ($GED = 2$)
2) Second, we randomly choose one sequence from SEQ, which is $seq_2 = \{op_2, op_1\}$. Therefore, in this case the random variable $S = 2$.
3) Third, the numbers of **RV** and **RE** operations in seq_2 are 0 and 2, respectively. Therefore, the random variables $X = 0$ and $Y = 2$ in this example.
4) Then, after conducting operations in seq_2, the number of vertices covered by relabelled edges is 3, and the number of vertices either relabelled or covered by relabelled edges is 3. Therefore, the random variables $Z = 3$ and $R = 3$.
5) Finally, GBD is considered to be the random variable, where $GBD = 2$ in this example.

In this paper, we aim to infer the probability distribution of GED when given GBD, which is essentially to calculate the following probability for given constants $\hat{\tau}$ and φ:

$$Pr[GED \leq \hat{\tau} \mid GBD = \varphi] = \sum_{\tau=0}^{\hat{\tau}} Pr[GED = \tau \mid GBD = \varphi] \quad (3)$$

By applying Bayes' Rule, we have:

$$Pr[GED \leq \hat{\tau} \mid GBD = \varphi] = \sum_{\tau=0}^{\hat{\tau}} \Lambda_1 \cdot \frac{\Lambda_3}{\Lambda_2}, \quad (4)$$

where

$$\Lambda_1(G'_1, G'_2; \tau, \varphi) = Pr[GBD = \varphi \mid GED = \tau] \quad (5)$$

$$\Lambda_2(G'_1, G'_2; \varphi) = Pr[GBD = \varphi] \quad (6)$$

$$\Lambda_3(G'_1, G'_2; \tau) = Pr[GED = \tau] \quad (7)$$

Therefore, the problem to solve becomes calculating the values of Λ_1, Λ_2 and Λ_3 when given extended graphs G'_1 and G'_2, and constants τ and φ. In this following three subsections V-A, V-B and V-C, we illustrate how to utilize our model to calculate Λ_1, Λ_2 and Λ_3, respectively.

A. Calculating $\mathbf{\Lambda_1}$

In this subsection, we aim to calculate $\Lambda_1 = Pr[GBD = \varphi \mid GED = \tau]$. The key idea to calculate Λ_1 is to expand its formula by applying Chain Rule [24] according to the dependency relationship between random variables in our model, where the expanded formula is:

$$\Lambda_1 = \sum_x \Omega_1 \sum_m \Omega_2 \sum_r \Omega_3 \cdot \Omega_4 \quad (8)$$

where

$$\Omega_1 = \frac{1}{N} \sum_s Pr[X = x, Y = \tau - x \mid S = s] \quad (9)$$

$$\Omega_2 = Pr\left[Z = m \mid Y = \tau - x\right] \quad (10)$$

$$\Omega_3 = Pr\left[GBD = \varphi \mid R = r\right] \quad (11)$$

$$\Omega_4 = Pr\left[R = r \mid X = x, Z = m\right] \quad (12)$$

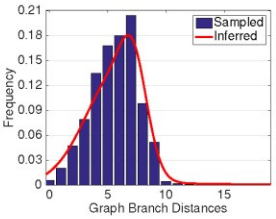

Fig. 5. Inferred prior distribution of GBDs on Fingerprint data set

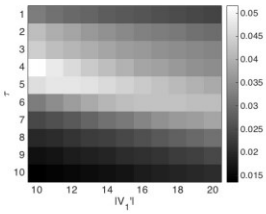

Fig. 6. Inferred prior distribution of GEDs on Fingerprint data set

Please refer to Appendix C for the closed forms of Ω_1, Ω_2, Ω_3, and Ω_4. Due to the space limitation, please refer to our technical report [23] for the proof of Equation (8).

B. Calculating $\mathbf{\Lambda_2}$

In this subsection, we aim to calculate Λ_2, which is essentially to infer the *prior distributions* of GBDs. In practice, we pre-compute the prior distribution of GBDs without knowing the query graph Q in the graph similarity search problem. This is because in most real-world scenarios [4] [15], the query graph Q ofter comes from the same population as graphs in database D. As a counter example, it is unusual to use a query graph of protein structure to search for similar graphs in social networks, and vice versa. Therefore, we assume that GBDs between Q and each graph G in database D, follow the same prior distributions as those among all graph pairs in D.

To calculate the prior distribution of GBDs, we first randomly sample $\alpha\%$ of graph pairs from the database D, and calculate the GBD between each pair of sampled graphs. Then, we utilize the *Gaussian Mixture Model* (GMM) [25] to approximate the distribution of GBDs between all pairs of sampled graphs, whose probability density function is:

$$f(\phi) = \sum_{i=1}^{K} \pi_i \cdot \mathcal{N}(\phi; \mu_i, \sigma_i) \quad (13)$$

where K is the number of components in the mixture model defined by user, and $\mathcal{N}$ is the probability density function of the normal distribution. Here, π_i, μ_i and σ_i are parameters of the i-th component in the mixture model, which can be inferred from the GBDs over sampled graph pairs. Please refer to [25] for the process of inferring parameters in GMM.

Finally, we can compute the prior probability $Pr[GBD = \varphi]$ by integrating the probability density function $f(\phi)$ on the adjacent interval of φ, i.e., $[\varphi - 0.5, \varphi + 0.5]$.

$$Pr[GBD = \varphi] = \int_{\varphi-0.5}^{\varphi+0.5} \sum_{i=1}^{K} \pi_i \cdot \mathcal{N}(\phi; \mu_i, \sigma_i)\, d\phi \quad (14)$$

Note that this integration technique is commonly used in the field called continuity correction [26], which aims to approximate discrete distributions by continuous distributions. In addition, the choice of integral interval $[\varphi - 0.5, \varphi + 0.5]$ is a common practice in the continuity correction field [27].

Example 5: We randomly sample 60,000 pairs of graphs from Fingerprint dataset of IAM Graph Database [28], where the distribution of GBDs between all pairs of sampled graphs is represented by the blue histogram in Figure 5. We infer the GBD prior distribution of sampled graph pairs, which is represented by the red line in Figure 5. This way, we can compute and store the probability $Pr[GBD = \varphi]$ for each possible value of variable φ by Equation (14), where the range of variable φ is analyzed in Section VI.

C. Calculating $\mathbf{\Lambda_3}$

In this subsection, we focus on calculating Λ_3, i.e., the prior distribution of GEDs. Recall that the GED computation is NP-

hard [4]. Therefore, sampling some graph pairs and calculating the GED between each pair is infeasible especially for large graphs, which means that we cannot simply infer the prior distribution of GEDs from sampled graph pairs.

To address this problem, we utilize the Jeffreys prior [29] as the prior distribution of GEDs. The Jeffreys prior is well-known for its non-informative property [30], which means that it provides very little additional information to the probabilistic model. Therefore, Jeffreys prior is a common choice in Bayesian methods when we know little about the actual prior distribution. Then, according to the definition of Jeffreys prior [29], the value of probability $Pr[GED = \tau]$ is calculated by:

$$Pr[GED = \tau]$$
$$= \frac{1}{C}\sqrt{\sum_{\varphi=0}^{2\tau} Pr[GBD = \varphi | GED = \tau] \cdot \mathcal{Z}^2(G'_1, G'_2; \tau, \varphi)} \quad (15)$$
$$= \frac{1}{C}\sqrt{\sum_{\varphi=0}^{2\tau} \Lambda_1(G'_1, G'_2; \tau, \varphi) \cdot \mathcal{Z}^2(G'_1, G'_2; \tau, \varphi)}, \quad (16)$$

where Equation (15) comes from the definition of the Jeffreys prior [29], which is the expected value of function $\mathcal{Z}^2$ with respect to the conditional distribution of variable GBD when given $GED = \tau$. Here, C is a constant for normalization, and the function $\mathcal{Z}$ is defined in the following Equation (17).

$$\mathcal{Z}(G'_1, G'_2; \tau, \varphi) = \left.\frac{\partial\{\log Pr[GBD|GED]\}}{\partial\{GED\}}\right|_{GED=\tau, GBD=\varphi} \quad (17)$$

where the symbol ∂ represents the partial derivative of a function, and the symbol $F(x)|_{x=k}$ means to substitute value k for variable x in the function $F(x)$. Please refer to Appendix C for the closed forms of Equation (16).

According to the closed form of Equation (16), we find that the value of probability $Pr[GED = \tau]$ only depends on the constant τ and the size of the extended graph G'_1, i.e., ($|V'_1|$). Therefore, for each data set, we pre-compute the value of function $Pr[GED = \tau]$ for each possible value of τ and $|V'_1|$, and store these pre-computed values in a matrix for quick looking up when searching similar graphs. In Section VI, we discuss the ranges of τ and $|V'_1|$ in detail. Note that, if there are k_1 possible values of τ and k_2 possible values of $|V'_1|$, then the normalization constant C in Equations (15) and (16) will be $C = 1/(k_1 \cdot k_2)$.

Example 6: We show part of the Jeffreys prior distribution of GEDs on Fingerprint data set [28] in Figure 6 as an example, where the x-axis represents values of $|V'_1|$, and the y-axis represents values of τ. The gray scale of each 1×1 block in Figure 6 represents the corresponding value of $Pr[GED = \tau]$.

VI. Graph Similarity Search with the Probabilistic Model

In this section, we first elaborate on our graph similarity search algorithm (i.e., GBDA) based on the model derived in the previous section, which consists of two stages: offline pre-processing and online querying. Then, we study the time and space complexity of these two stages in detail.

A. Graph Similarity Search Algorithm

Given a query graph Q, a graph database D, a similarity threshold $\hat{\tau}$, and a probability threshold γ, the search results D_0 is achieved by Algorithm 1, where Q' and G' are the extended graphs of Q and G, respectively. Step 1 tagged with symbol * is pre-computed in the offline stage, as discussed in Sections V-B and V-C. We give the Example 7 below to better illustrate the process of Algorithm 1.

Algorithm 1 Graph Similarity Search with Graph Branch Distance Approximation (GBDA)

Input: a query graph Q, a graph database D, a similarity threshold $\hat{\tau}$, and a probability threshold γ
Output: the search result D_0
for each graph $G \in D$ **do**
  Step 1*: Pre-compute Λ_2 and Λ_3 for all possible inputs
  Step 2: Calculate $GBD(Q', G')$ by Definition 4.
  Step 3: Given $GBD(Q', G') = \varphi$, we calculate
  $$\Phi = Pr[GED(Q, G) \leq \hat{\tau} \mid GBD(Q, G) = \varphi] = \sum_{\tau=0}^{\hat{\tau}} \Lambda_1(Q', G'; \tau, \varphi) \cdot \frac{\Lambda_3(Q', G'; \tau)}{\Lambda_2(Q', G'; \varphi)}$$
  where $\Lambda_1(Q', G'; \tau, \varphi)$ is calculated by Equation (8).
  Step 4: Insert G into D_0 if $\Phi \geq \gamma$
end for

Example 7: Assume that the graph G_1 in Figure 1 is the query graph Q, and G_2 in Figure 1 is a graph in database D. Given the similarity threshold $\hat{\tau} = 3$ and the probability threshold $\gamma = 0.8$, the process of determining whether G_2 should be in the search result D_0 is as follows.

1) First, we pre-compute $\Lambda_2(Q', G'_2; \varphi)$ and $\Lambda_3(Q', G'_2; \tau)$ by inferring the prior distributions of GBDs and GEDs on database D, respectively. Since this is a simulated example and there is no concrete database D, we assume that $\Lambda_3(Q', G'_2; \tau)/\Lambda_2(Q', G'_2; \varphi) \equiv 0.8$ for all possible values of τ and φ.
2) Second, from Example 2, we know $GBD(Q, G_2) = 3$.
3) Then, according to Equation (8), we can calculate
   $\Phi = \sum_{\tau=0}^{\hat{\tau}} \Lambda_1(Q', G'_2; \tau, \varphi) \cdot \frac{\Lambda_3(Q', G'_2; \tau)}{\Lambda_2(Q', G'_2; \varphi)}$
   $= (0 + 0 + 0.5113 + 0.5631) \times 0.8 = 0.8595 > \gamma = 0.8$
4) Therefore, G_2 is inserted into the search result D_0.

B. Complexity Analysis of Online Stage

The online querying stage in our approach includes Steps 2, 3 and 4 in Algorithm 1, where Step 4 clearly costs $O(1)$ time for each graph G. In addition, from the discussions in Section III, Step 2 costs $O(nd)$ time, where $n = \max\{|V_G|, |V_Q|\}$, and d is the average degree of graph G.

In Step 3, since Λ_2 and Λ_3 have already been pre-computed in the offline stage (i.e., Step 1), their values can be obtained in $O(1)$ time for each $\tau \in [0, \hat{\tau}]$ and graph $G \in D$.

Now we focus on analyzing the time complexity of computing Λ_1 in Step 3 of Algorithm 1. Let the time for computing Ω_1, Ω_2, Ω_3 and Ω_4 in Equation (8) be C_1, C_2, C_3 and C_4, respectively. According to Equation (8), the time for computing Λ_1 for each $\tau \in [0, \hat{\tau}]$ and graph $G \in D$ is:

$$\overbrace{xC_1}^{\sum\Omega_1} + \overbrace{xmC_2}^{\sum\Omega_2} + \overbrace{xmr(C_3 \cdot C_4)}^{\sum\Omega_3 \cdot \Omega_4} \quad (18)$$

where x, m and r are the summation subscripts in Equation (8), and the ranges of x, m and r are:

- $x \in [0, \tau]$. Since x is the number of **RV** operations, x must not be larger than the number of graph edit operations τ.
- $m \in [0, 2\tau]$. Since m is the number of vertices covered by relabelled edges given the relabelled edge number $Y = \tau - x$, and each edge can cover at most two vertices, we have $0 \leq m \leq 2(\tau - x) \leq 2\tau$.
- $r \in [0, 3\tau]$. Note that r is the number of vertices either relabelled or covered by relabelled edges when the relabelled vertex number is $X = x$ and the number of vertices covered by relabelled edges is $Z = m$. Therefore, we have $0 \leq r \leq x + m \leq \tau + 2\tau = 3\tau$.

In addition, according to the closed form of Equation (8) in Appendix C, it is clear that $C_1 = C_3 = C_4 = O(1)$ and $C_2 = O(m) = O(\tau)$. According to Equation (18), the time of computing Λ_1 for each $\tau \in [0, \hat{\tau}]$ and graph $G \in D$ is:

$$\overbrace{O(\tau)}^{\sum \Omega_1} + \overbrace{O(\tau^3)}^{\sum \Omega_2} + \overbrace{O(\tau^3)}^{\sum \Omega_3 \cdot \Omega_4} = O(\tau^3) \quad (19)$$

Moreover, from the above discussions about the ranges of summation subscripts, for any $\tau \in [0, \hat{\tau})$, we have:

$$\begin{aligned}
&\textstyle\sum_{x=0}^{\hat{\tau}} \sum_{m=0}^{2\hat{\tau}} \Omega_2(m, x, \hat{\tau}) \\
=&\textstyle\sum_{x=0}^{\hat{\tau}} \sum_{m=0}^{2\hat{\tau}} Pr[Z = m \mid Y = \hat{\tau} - x] \quad (20) \\
=&\textstyle\sum_{x=0}^{\tau} \sum_{m=0}^{2\tau} Pr[Z = m \mid Y = \tau - x] \\
&+ \textstyle\sum_{x=0}^{\hat{\tau}-\tau-1} \sum_{m=0}^{2\tau} Pr[Z = m \mid Y = \hat{\tau} - x] \\
&+ \textstyle\sum_{x=0}^{\tau} \sum_{m=2\tau+1}^{2\hat{\tau}} Pr[Z = m \mid Y = \hat{\tau} - x] \\
&+ \textstyle\sum_{x=0}^{\hat{\tau}-\tau-1} \sum_{m=2\tau+1}^{2\hat{\tau}} Pr[Z = m \mid Y = \hat{\tau} - x] \quad (21) \\
=&f(m, x, \hat{\tau}) + \textstyle\sum_{x=0}^{\tau} \sum_{m=0}^{2\tau} \Omega_2(m, x, \tau) \quad (22)
\end{aligned}$$

where $f(m, x, \hat{\tau})$ is sum of last three terms in Equation (21). Note that Equation (21) is a sum on four disjoint two-dimensional intervals whose combination is the sum interval of Equation (20).

Equation (22) means, the value of $\sum_{x,m} \Omega_2(m, x, \tau)$ where $\tau < \hat{\tau}$ have already been calculated in the process of computing $\sum_{x,m} \Omega_2(m, x, \hat{\tau})$. Therefore, we can reduce redundant computations by only computing $\sum_{x,m} \Omega_2(m, x, \hat{\tau})$ once to obtain values of $\sum_{x,m} \Omega_2(m, x, \tau)$ for all $\tau < \hat{\tau}$. Similar conclusions can also be derived for $\sum_{x,m,r} \Omega_3(r, \varphi) \cdot \Omega_4(x, r, m)$, where the detailed proofs are omitted here.

Therefore, the time cost of Step 3 in Algorithm 1 is:

$$\overbrace{O(\hat{\tau}^3)}^{\sum \Omega_2 \sum \Omega_3 \cdot \Omega_4} + \textstyle\sum_{\tau=0}^{\hat{\tau}} \{\overbrace{O(\tau)}^{\sum \Omega_1} + \overbrace{O(1)}^{\Lambda_2}\} = O(\hat{\tau}^3) \quad (23)$$

for each graph G in database D.

Finally, we can obtain Theorem 3.

Theorem 3: The time of the whole online stage is:

$$\overbrace{O(nd)}^{\text{Step 2}} + \overbrace{O(\hat{\tau}^3)}^{\text{Step 3}} + \overbrace{O(1)}^{\text{Step 4}} = O(nd + \hat{\tau}^3) \quad (24)$$

for each graph G in database D, where $n = \max\{|V_G|, |V_Q|\}$, d is the average degree of graph G, and $\hat{\tau}$ is the similarity threshold in the graph similarity search problem.

Proof: Please refer to the discussions above. ■

Note that the similarity threshold $\hat{\tau}$ is often set as a small value (i.e., $\hat{\tau} \leq 10$) and does not increase with the number of vertices n in previous studies [4] [15], thus, we can assume that $\hat{\tau}$ is a constant with regard to n when the graph is sufficiently large. Moreover, most real-world graphs studied in related works [11] [12] are *scale-free graphs* [16], whose average degrees $d = O(\log n)$ as proved in our technical report [23].

C. Complexity Analysis of Offline Stage

The offline pre-processing stage in our approach is Step 1 in Algorithm 1, which is essentially to pre-compute the prior distributions of GEDs and GBDs respectively among all pairs of graphs involved in the graph similarity search.

1) Complexity Analysis of Computing the Prior Distribution of GBDs: As discussed in Section 5.1, the prior distributions of GBDs can be pre-computed by the following four steps:

Step 1.1: Sample N graph pairs from the database D.
Step 1.2: Calculate GBD between each sampled graph pairs.
Step 1.3: Learn the Gaussian Mixture Model (GMM) of the GBDs between sampled graph pairs.
Step 1.4: Calculate $Pr[GBD = \varphi]$ for each possible value of φ by using Equation (14).

It is clear that Step 1.1 costs $O(N)$ time. From the discussions in Section III, Step 1.2 costs $O(N \cdot nd)$ time, where n is the maximal number of vertices among the sampled graphs, and d is the average degree of the sampled graphs. The learning process of GMM in Step 1.3 costs $O(N \cdot K\epsilon)$ time [25], where K is the number of components in GMM, and ϵ is the maximal learning iterations for learning GMM.

As for Step 1.4, since φ is the value of GBD, from the definition of GBD, the possible values of φ are essentially $\{0, 1, 2..., n\}$, where n is the maximal number of vertices among the sampled graphs. According to Equation (14), computing $Pr[GBD = \varphi]$ for each φ costs $O(K)$ time, where K is the number of components in GMM derived in Step 1.3. Thus, Step 1.4 costs $O(nK)$ time.

Note that in the Gaussian Mixture Model, the component number K and the maximal learning iterations ϵ are fixed constants. Therefore, the prior distributions of GBDs can be calculated in time

$$\overbrace{O(N)}^{\text{Step 1.1}} + \overbrace{O(Nnd)}^{\text{Step 1.2}} + \overbrace{O(NK\epsilon)}^{\text{Step 1.3}} + \overbrace{O(nK)}^{\text{Step 1.4}} = O(Nnd) \quad (25)$$

where N is the number of graph pairs sampled in Step 1.1, n is the maximal number of vertices among sampled graphs, and d is the average degree of sampled graphs.

Based on the discussions above, the number of pre-computed values of $Pr[GBD = \varphi]$ is at most n. Thus, the space cost of storing the prior distribution of GBDs is $O(n)$.

2) Complexity Analysis of Computing the Prior Distribution of GEDs: According to the discussions in Section V-C, computing the prior distribution of GEDs is essentially calculating Equation (16) for each possible values of τ and $|V_1'|$.

First, from the closed form of Equation (16) in Appendix C, when τ and $|V_1'|$ are fixed values, it is clear that computing $\frac{d}{d\tau} \log \Lambda_1$ costs the same time as computing Λ_1, which is $O(\hat{\tau}^3)$, where $\hat{\tau}$ is the user-defined similarity threshold. n and d are the maximal number of vertices and the average degree among all graphs, respectively.

Then, since one graph edit operation can at most change two branches, when the GED between two graphs is τ, the possible

TABLE III
STATISTICS OF DATA SETS

Data Set	$\|D\|$	$\|\mathcal{Q}\|$	$\mathcal{V}_m$	$\mathcal{E}_m$	d	Scale-free
AIDS	1896	100	95	103	2.1	Yes
Finger	2159	114	26	26	1.7	Yes
GREC	1045	55	24	29	2.1	Yes
AASD	37995	100	93	99	2.1	Yes
Syn-1	3430	70	100K	1M	9.6	Yes
Syn-2	3430	70	100K	1M	9.4	No

Note: $|D|$ is the number of graphs in database D. $|\mathcal{Q}|$ is the number of query graphs. $\mathcal{V}_m$ and $\mathcal{E}_m$ are the maximal numbers of vertices and edges, respectively, while d is the average degree. K means thousand and M means million.

GBD values (i.e., φ in Equation (16)) between these two graphs are $\{0, 1, 2..., 2\tau\}$. Therefore, according to Equation (16), the prior probability value of $Pr[GED = \tau]$ can be calculated in time complexity $O(2\tau \cdot \tau^3) = O(\tau^4)$ when τ and $|V_1'|$ are fixed values.

Finally, recall that computing the GED prior distribution is essentially calculating Equation (16) for all possible values of τ and $|V_1'|$, and it is clear that the possible values of τ are $\{0, 1, 2..., \hat{\tau}\}$, where $\hat{\tau}$ is the user-defined similarity threshold. In addition, the possible values of $|V_1'|$ are essentially $\{1, 2, ...n\}$, where n is the maximal number of vertices among all graphs involved the graph similarity search. Therefore, the time of calculating the GED prior distribution is:

$$O(\hat{\tau} \cdot n \cdot \hat{\tau}^4) = O(n\hat{\tau}^5)$$

According to Section V-C, we need to store a matrix whose rows represent possible values of τ, and columns represent possible values of $|V_1'|$. Therefore, the space cost of storing the prior distribution of GEDs is $O(\hat{\tau} \cdot n)$.

Finally, we have Theorem 4.

Theorem 4: The time complexity of the offline stage is:

$$\overbrace{O(Nnd)}^{\text{GBD Prior}} + \overbrace{O(n\hat{\tau}^5)}^{\text{GED Prior}} = O(Nnd + n\hat{\tau}^5)$$

and the space complexity of the offline stage is:

$$\overbrace{O(n)}^{\text{GBD Prior}} + \overbrace{O(\hat{\tau} \cdot n)}^{\text{GED Prior}} = O(n(1 + \hat{\tau}))$$

where N is the number of graph pairs sampled in Step 1.1, n and d are the maximal number of vertices and the average degree among all the graphs involved in the graph simialrity search, respetively. $\hat{\tau}$ is the user-defined similarity threshold.

Proof: Please refer to the discussions above. ■

VII. EXPERIMENTS

A. Data Sets and Settings

We first present the experimental data sets for evaluating our approaches. There are 4 real-world data sets (i.e., AIDS, Fingerprint and GREC from the IAM Graph Database [28], and AIDS Antiviral Screen Data (AASD) [31]), and 2 synthetic data sets (i.e., Syn-1 and Syn-2). The details about the data sets are listed in Table III.

The 4 real-world data sets are widely-used for evaluating the performance of GED estimation methods in previous works [11] [12] [13]. In order to evaluate how well the GED is approximated, we must know the exact value of GED, which is NP-hard to compute [4]. Specifically, even the state-of-the-art method [6] cannot compute an exact GED for graphs with 100 vertices within 48 hours on our machine (with 32 Intel E5 2-core, 2.40 GHz CPUs and 128GB DDR3 RAMs).

TABLE IV
COSTS OF COMPUTING GBD PRIOR DISTRIBUTION

Data Set	AIDS	Finger	GREC	AASD	Syn-1	Syn-2
Time Costs	11.1s	7.5s	20.6s	232.4s	3.8h	3.2h
Space Costs	0.06kb	0.04kb	0.10kb	1.21kb	13.3gb	0.3gb

TABLE V
COSTS OF COMPUTING GED PRIOR DISTRIBUTION

Data Set	AIDS	Finger	GREC	AASD	Syn-1	Syn-2
Time Costs	70.32h	16.91h	15.40h	69.16h	6.31h	6.31h
Space Costs	1.5kb	0.4kb	0.4kb	1.4kb	0.1kb	0.1kb

Note: h means hours and s means seconds. kb means KBytes and gb means GBytes.

However, we still manage to evaluate our proposed method on large graphs. To address the problem above, we generate 2 sets of large random graphs (i.e., Syn-1 and Syn-2), where the GED between each pair of graphs is known. Both data sets Syn-1 and Syn-2 contain 7 subsets of graphs, where each subset contains 500 graphs whose numbers of vertices are 1K, 2K, 5K, 10K, 20K, 50K, and 100K, respectively. The difference between data sets Syn-1 and Syn-2 is that the graphs in Syn-1 satisfy the *scale-free* property [16] while graphs in Syn-2 are not. The algorithm of generating synthetic graphs with known GEDs is described in our technical report [23].

Note that, the scale-free property of real data sets is testified by checking whether the degree distributions of the vertices in real data sets follow the power-law distribution, while the scale-free property of Syn-1 data set and the non-scale-free property of Syn-2 data sets are ensured by our algorithm of generating synthetic graphs in our technical report [23].

For each real data set, we randomly select 5% graphs as query graphs, while the remaining 95% graphs constitute the graph database D. For each synthetic data set, we randomly select 10 graphs from each of its subset as query graphs.

On real data sets, we evaluate our method with the similarity thresholds $\hat{\tau} = \{1, 2, ..., 10\}$, which are commonly-used values of the similarity thresholds in previous studies [4] [15]. On the synthetic data sets, we test our method with larger similarity thresholds $\hat{\tau} = \{10, 11, 12, ..., 30\}$ to show that, when the similarity threshold is larger than the commonly-used values (i.e., $\{1, 2, ..., 10\}$), our GBDA method is still more efficient than the competitors on large graphs.

B. Evaluating Offline Stage

In this subsection, we evaluate the time and space costs of the offline stage in our GBDA approach, which is essentially to pre-compute the prior distributions of GEDs and GBDs, on both real and synthetic data sets. Tables IV and V present the time and space costs of estimating GBD and GED prior distributions on different data sets, respectively, where the number of graph pairs sampled to estimate the GBD prior distribution is set to $N = 100,000$.

The experimental results generally confirm the complexity analysis in Section VI-C. Specifically, since the number of sampled graph pairs N and the similarity threshold $\hat{\tau}$ are fixed values in our experiments, the cost of inferring the GBD prior distribution grows with n and d, while the cost of estimating the GED prior distribution depends on n, where n and d are the maximal number of vertices and the average degree among all the graphs involved in the graph simialrity search, respetively.

Note that, the costs of computing the GED prior distribution on *synthetic* graphs do not exactly follow the theoretical analysis. This is because the numbers of vertices in synthetic graphs

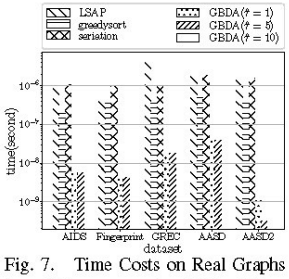

Fig. 7. Time Costs on Real Graphs

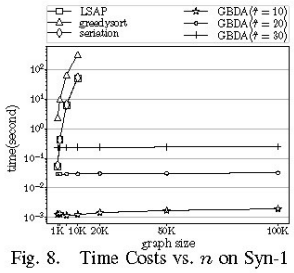

Fig. 8. Time Costs vs. n on Syn-1

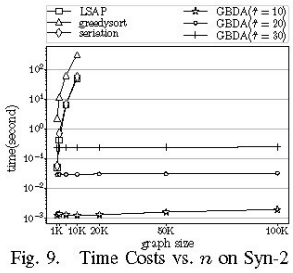

Fig. 9. Time Costs vs. n on Syn-2

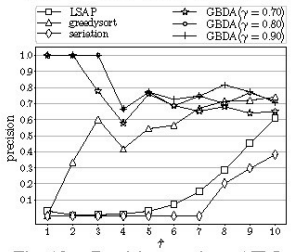

Fig. 10. Precision vs. $\hat{\tau}$ on AIDS

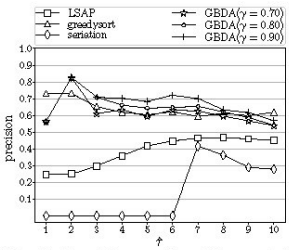

Fig. 11. Precision vs. $\hat{\tau}$ on Fingerprint

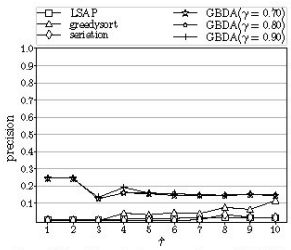

Fig. 12. Precision vs. $\hat{\tau}$ on GREC

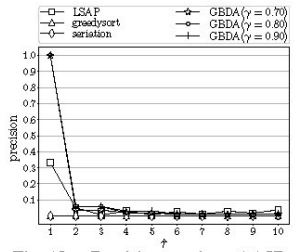

Fig. 13. Precision vs. $\hat{\tau}$ on AASD

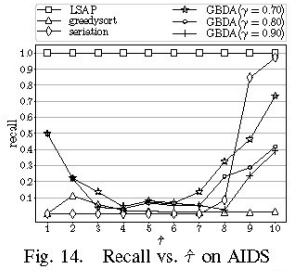

Fig. 14. Recall vs. $\hat{\tau}$ on AIDS

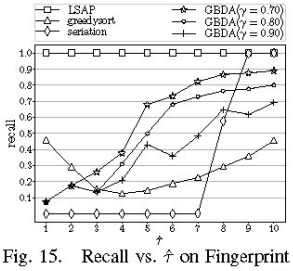

Fig. 15. Recall vs. $\hat{\tau}$ on Fingerprint

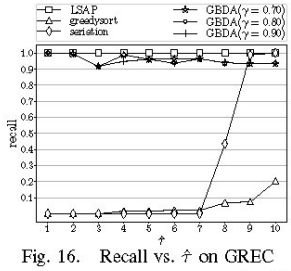

Fig. 16. Recall vs. $\hat{\tau}$ on GREC

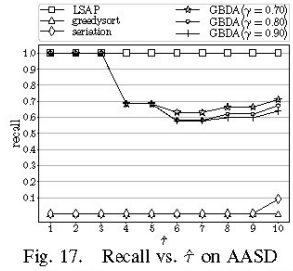

Fig. 17. Recall vs. $\hat{\tau}$ on AASD

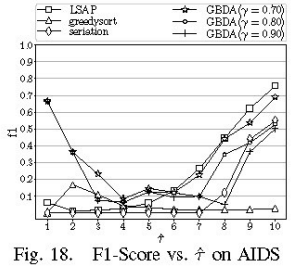

Fig. 18. F1-Score vs. $\hat{\tau}$ on AIDS

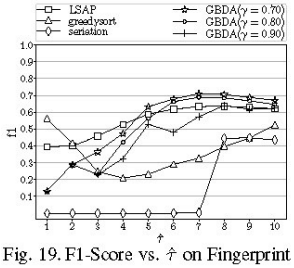

Fig. 19. F1-Score vs. $\hat{\tau}$ on Fingerprint

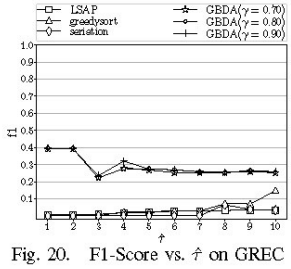

Fig. 20. F1-Score vs. $\hat{\tau}$ on GREC

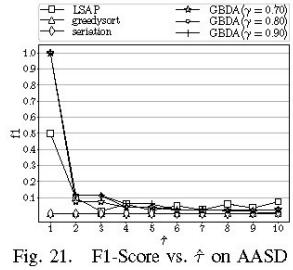

Fig. 21. F1-Score vs. $\hat{\tau}$ on AASD

have only 7 possible values (i.e., 1K, 2K, 5K, 10K, 20K, 50K, and 100K), instead of the worst-case range $1 \sim 100K$ as discussed in Section VI-C. In addition, although the synthetic graphs are larger than the real ones, the number of possible values of n on synthetic graphs are smaller than that on real graphs, where n is the number of vertices in graphs. Therefore, the costs of computing the GED prior distribution on synthetic graphs are smaller than the costs on real data sets.

C. Evaluating Online Stage

In this subsection, we compare the efficiency and accuracy of the online stage of our GBDA approach with three competitors (i.e., LSAP [11], Greedy-Sort-GED [12] and Graph Seriation [13]), by conducting graph similarity search tasks over both real and synthetic data sets. In addition, we analyze how the efficiency and effectiveness (i.e., accuracy, recall and F1-score) of our method are influenced by the parameters, such as the similarity threshold $\hat{\tau}$, the probability threshold γ, and the number, n, of vertices.

1) The Efficiency Evaluation: In this subsection, we evaluate the query efficiency of our GBDA approach and three competitors on real and synthetic data sets. Note that the time cost of our GBDA methods depends on both n and $\hat{\tau}$, while the competitors' time costs only depend on n, where n is the number of vertices in graphs and $\hat{\tau}$ is the similarity threshold. Therefore, the experiments in this subsection are conducted under a fixed probability threshold $\gamma = 0.9$, since γ does not affect the time costs of all the methods.

Specifically, given a specific method and its parameters (e.g., $\hat{\tau}$ and γ), for each query graph Q, we utilize this method to obtain a set of graphs similar to graph Q from each data set, and we record the average query response time for each data set, which are presented in Figures 7∼9. Particularly, for a specific method under one specific parameter set (e.g., $\hat{\tau}$ and γ), each query's response time is recorded and counted only once for each data set, and we present the average of response times for all queries in the experimental figures.

The result in Figure 7 shows that our GBDA approach is more efficient than the three competitors on all real data sets where $\hat{\tau}$ is set to 1, 5 and 10, respectively. In addition, we studied how the number, n, of vertices in graphs influences

the efficiency of our GBDA approach by comparing the query response time on synthetic data sets with various similarity thresholds $\hat{\tau}$, where the results are shown in Figures 8 and 9. The results show that our GBDA approach is more efficient than the competitors on both scale-free and non-scale-free graphs, where the similarity threshold $\hat{\tau} \leq 20$. Particularly, when the similarity threshold $\hat{\tau} = 30$, although our GBDA approach costs more time than the other methods on graphs with 1,000 vertices, our approach is faster than the competitors on larger graphs with more than 2,000 vertices. Therefore, when the similarity threshold is larger than the commonly-used values (i.e., $\{1, 2, ..., 10\}$), the time cost of our algorithm is still smaller than the compared methods on large graphs (i.e., graphs with more than 2,000 vertices).

Note that the competitors (i.e., LSAP, Greedy-Sort-GED and Graph Seriation) can handle graphs with at most 20K vertices on our machine. Specifically, when the graphs have more than 20K vertices, all the competitors consume more than 128 GB memory on our machine, which exceeds the capacity of the physical memory. However, our GBDA method can handle graphs with 100K vertices efficiently, which confirms that our method has better scalability (with respect to the number, n, of vertices) than the competitors.

2) The Effectiveness Evaluation: We evaluate the effectiveness of our GBDA approach and three competitors by comparing the precision, recall, and F1-score [32] of the query results on each real data set with probability thresholds $\gamma = 0.7, 0.8$ and 0.9, respectively.

The results in Figures 10∼13 show that our approach always outperforms the other three competitors in precision on AIDS, Fingerprint and GREC data sets, and achieves the highest precisions among all methods where $\hat{\tau} = 1, 2, 3, 4, 5$ and 7 on AASD data set. The results in Figures 14∼17 show that our method has the second highest recalls under most parameter settings. Note that, LSAP method returns a lower bound of GED [11], and therefore the recall of its search result is always 100%. However, if we evaluate the methods by the F1-score, our method always outperforms the other three competitors on AIDS, Fingerprint and GREC data sets, and achieves the highest F1-scores among all methods when $\hat{\tau} = 1, 2, 3, 4, 5$ and 7 on AASD data set. Therefore, the experimental results confirm that the effectiveness of our method outperforms the competitors' under most parameter settings on the real data sets in our experiments.

In addition, we study how the number, n, of vertices in graphs influences the effectiveness (i.e., precision, recall and F1-score) of our approach on the Syn-1 data set with various probability thresholds γ and similarity thresholds $\hat{\tau}$ in our technical report [23] due to the space limitation.

The results in our technical report [23] show that, the precision of our method outperforms the competitors on Syn-1 data set, where the similarity threshold $\hat{\tau} = 15, 20, 25$ and 30. Moreover, there is no significant difference between the precision of our method under various settings of the probability threshold γ, which demonstrates the robustness of our method under different settings of parameters. The recalls of our method are slightly lower than the LSAP method, but much higher than the Greedy-Sort-GED and Graph Seriation methods, where the similarity threshold $\hat{\tau} = 20, 25$ and 30. Finally, the F1-Scores of our method are mostly higher than the competitors. Therefore, the experimental results on synthetic graphs demonstrate that our method is more effective than the competitors on large graphs.

D. Comparing with Alternatives

In this subsection, we compare the effectiveness of our GBDA approach and two variants of our method by comparing the F1-score [32] of the query results on each real data set with probability thresholds $\gamma = 0.9$. The variants of our GBDA method, i.e., GBDA-V1 and GBDA-V2, are considered as alternatives of GBDA method, which are illustrated as follows.

GBDA-V1: The method GBDA-V1 utilizes the average number of vertices among a sample of graphs from the graph database as the parameter $|V_1'|$ when computing Λ_1 and Λ_3 in Algorithm 1, instead of using the number of vertices in the extended query graph Q' as the parameter $|V_1'|$.

GBDA-V2: The method GBDA-V2 exploits the variant GBD (VGBD) instead of the original GBD value when computing Λ_1 and Λ_2 in Algorithm 1, where VGBD is defined as:

$$VGBD(G_1, G_2) = \max\{|V_1|, |V_2|\} - w \cdot |B_{G_1} \cap B_{G_2}| \quad (26)$$

where B_{G_1} and B_{G_2} are the multisets of all branches in graphs G_1 and G_2, respectively, $|V_1|$ and $|V_2|$ are numbers of vertices in graphs G_1 and G_2, respectively, and w is a user-defined constant.

Specifically, we denote the number of sampled graphs in method GBDA-V1 by α, and we test the method GBDA-V1 by setting $\alpha = 10$, 50, and 100 on real data sets. In addition, we evaluate the effectiveness of method GBDA-V2 on real data sets by setting the parameter $w = 0.1$ and 0.5, where w is defined in Equation (26).

The results in Figures 22∼25 show that our GBDA method achieves higher F1-score than its variant GBDA-V1 for the similarity threshold $\hat{\tau} \leq 4$, but generally has the same F1-score as GBDA-V1 for $\hat{\tau} \geq 5$. In addition, from the results in Figures 26∼29, we can find that the F1-score of our GBDA method is higher than or almost the same as the GBDA-V2 method for the similarity threshold $\hat{\tau} \leq 2$ on all real data sets, and slightly lower than GBDA-V2 method on Fingerprint data set for $\hat{\tau} \geq 3$. In general, the experimental results show that our GBDA method outperforms methods GBDA-V1 and GBDA-V2 in most cases for similarity threshold $\hat{\tau} \leq 5$, and performs better or almost the same as GBDA-V1 and GBDA-V2 methods for the similarity threshold $\hat{\tau} \geq 6$.

VIII. Related Works

A. Exact GED Computation

The state-of-the-art method for exact GED computation is the A^* algorithm [5] and its variant [6], whose time costs are exponential with respect to graph sizes [4]. To address this NP-hardness problem, most graph similarity search method are based on the filter-and-verification framework [4] [18] [19] [15], which first filters out undesirable graphs from the graph databases and then only verifies the remaining candidates. A

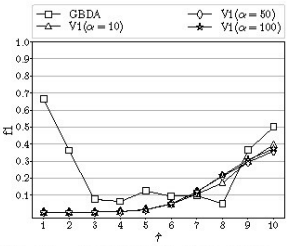

Fig. 22. F1-Score vs. $\hat{\tau}$ on AIDS (Compared with GBDA-V1)

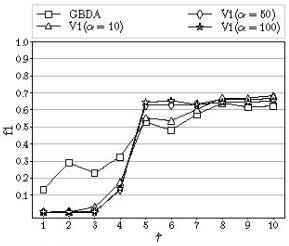

Fig. 23. F1-Score vs. $\hat{\tau}$ on Fingerprint (Compared with GBDA-V1)

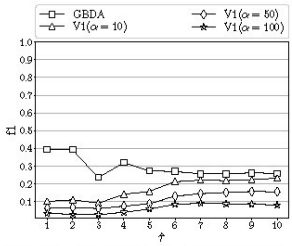

Fig. 24. F1-Score vs. $\hat{\tau}$ on GREC (Compared with GBDA-V1)

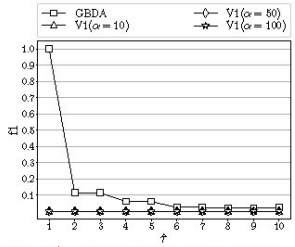

Fig. 25. F1-Score vs. $\hat{\tau}$ on AASD (Compared with GBDA-V1)

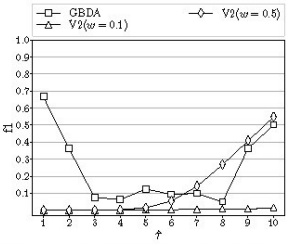

Fig. 26. F1-Score vs. $\hat{\tau}$ on AIDS (Compared with GBDA-V2)

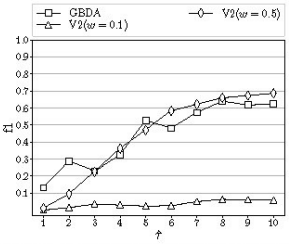

Fig. 27. F1-Score vs. $\hat{\tau}$ on Fingerprint (Compared with GBDA-V2)

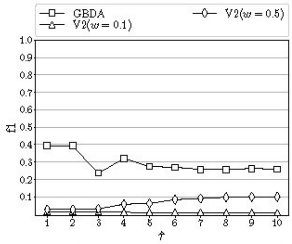

Fig. 28. F1-Score vs. $\hat{\tau}$ on GREC (Compared with GBDA-V2)

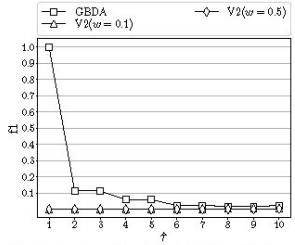

Fig. 29. F1-Score vs. $\hat{\tau}$ on AASD (Compared with GBDA-V2)

common filtering approach is to use the distance between sub-structures of two graphs as a lower bound of their GED, which includes tree-based [18], path-based [33] , branch-based [15] and partition-based [34] approaches. In this paper, we adopt the branch structure [15] to build our model. However, we re-define the distance between branches, since the original definition [15] of branch distances requires $O(n^3)$ time for computation while ours only requires $O(nd)$ time. In addition, a recent paper [35] propose a multi-layer indexing approach to accelerate the filtering process based on their proposed partition-based filtering method.

B. GED Estimation

In this paper, we focus on GED estimation approaches. One well-studied method [10] [11] is to utilize the solution of a *linear sum assignment problem* (LSAP) as an estimation of GED. The LSAP is an optimization problem which can be exactly solved by Hungarian method [36], or be approximately solved by the greedy method [12] and the genetic algorithm [37]. In our experiment, we compare our GBDA method with the exact [11] and greedy [12] solutions of LSAP. Since the exact solution of LSAP defined on two graphs is a lower bound of their GED [11], the LSAP method can always obtain all graphs whose GED to the query graph is no larger than the similarity threshold and achieve 100% recall in the similarity search tasks. On the other hand, the Greedy-Sort-GED method [12] solves LSAP approximately and has no bound to actual GED. However, the Greedy-Sort-GED method generally achieves better estimations of GEDs [12] and higher precisions in graph similarity search as shown in our experiments. Another state-of-the-art approach compared in our experiment is *graph seriation* [13], which first converts graphs into one-dimensional vectors by extracting their leading eigenvalues of the adjacency matrix, and then exploits a probabilistic model based on these vectors to estimate the GED. Although both graph seriation and our approach utilize probabilistic models, the structure of our model is totally different from the prior work [13]. In addition, their model takes the leading eigenvalues of the adjacency matrix as the inputs, while the inputs of our model are the GBDs. Moreover, our GBDA method outperforms the competitors' (i.e., the LSAP [11], Greedy-Sort-GED [12] and Graph Seriation [13]) under most parameter settings on real data sets.

IX. Conclusions

In this paper, we define the branch distance between two graphs (GBD), and further prove that the GBD has a probabilistic relationship with the GED by considering branch variations as the result ascribed to graph edit operations and modeling this process by probabilistic approaches. Furthermore, this relation between GED and GBD is leveraged to perform graph similarity searches. Experimental results demonstrate both the correctness and effectiveness of our approach, which outperforms the comparable methods.

Acknowledgments

The work is partially supported by the Hong Kong RGC GRF Project 16214716, National Grand Fundamental Research 973 Program of China under Grant 2014CB340303, the National Science Foundation of China (NSFC) under Grant No. 61729201, Science and Technology Planning Project of Guangdong Province, China, No. 2015B010110006, Huawei Co. Ltd Collaboration Project, YBCB2009041-45, Microsoft Research Asia Collaborative Research Grant, NSF OAC No.1739491, Lian Startup No.220981 Kent State University.

References

[1] A. Sanfeliu and K.-S. Fu, "A distance measure between attributed relational graphs for pattern recognition," *IEEE transactions on systems, man, and cybernetics*, no. 3, pp. 353–362, 1983.

[2] T. Caelli and S. Kosinov, "An eigenspace projection clustering method for inexact graph matching," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 4, pp. 515–519, 2004.

[3] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Learning Theory and Kernel Machines*. Springer, 2003, pp. 129–143.

[4] Z. Zeng, A. K. Tung, J. Wang, J. Feng, and L. Zhou, "Comparing stars: on approximating graph edit distance," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 25–36, 2009.

[5] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[6] K. Gouda and M. Hassaan, "Csi_ged: An efficient approach for graph edit similarity computation," in *32nd IEEE International Conference on Data Engineering, ICDE*, 2016, pp. 265–276.

[7] R. Ibragimov, M. Malek, J. Guo, and J. Baumbach, "Gedevo: an evolutionary graph edit distance algorithm for biological network alignment," in *OASIcs-OpenAccess Series in Informatics*, vol. 34. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[8] R. P. Ron Milo, *Cell Biology by the Numbers*. Garland Science, 2015.

[9] X. Gao, B. Xiao, D. Tao, and X. Li, "A survey of graph edit distance," *Pattern Analysis and applications*, vol. 13, no. 1, pp. 113–129, 2010.

[10] S. Bougleux, L. Brun, V. Carletti, P. Foggia, B. Gaüzère, and M. Vento, "Graph edit distance as a quadratic assignment problem," *Pattern Recognition Letters*, 2016.

[11] K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," *Image and Vision computing*, vol. 27, no. 7, pp. 950–959, 2009.

[12] K. Riesen, M. Ferrer, and H. Bunke, "Approximate graph edit distance in quadratic time," *IEEE/ACM transactions on computational biology and bioinformatics*, 2015.

[13] A. Robles-Kelly and E. R. Hancock, "Graph edit distance from spectral seriation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 3, pp. 365–378, 2005.

[14] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012, vol. 3.

[15] W. Zheng, L. Zou, X. Lian, D. Wang, and D. Zhao, "Graph similarity search with edit distance constraint in large graph databases," in *CIKM '13: Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013.

[16] Wikipedia, "Scale-free network," 2017, https://en.wikipedia.org/w/index.php?title=Scale-free_network.

[17] D. J. Hu, "Latent dirichlet allocation for text, images, and music," *University of California, San Diego. Retrieved April*, vol. 26, 2009.

[18] G. Wang, B. Wang, X. Yang, and G. Yu, "Efficiently indexing large sparse graphs for similarity search," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 3, pp. 440–451, 2012.

[19] X. Zhao, C. Xiao, X. Lin, and W. Wang, "Efficient graph similarity joins with edit distance constraints," in *IEEE 28th International Conference on Data Engineering*. IEEE, 2012, pp. 834–845.

[20] C. S. Library, "std::lexicographical_compare," 2017, http://www.cplusplus.com/reference/algorithm/lexicographical_compare/.

[21] D. Justice and A. Hero, "A binary linear programming formulation of the graph edit distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1200–1214, 2006.

[22] F. Serratosa, "Fast computation of bipartite graph matching," *Pattern Recognition Letters*, vol. 45, pp. 244–250, 2014.

[23] Z. Li, X. Jian, X. Lian, and L. Chen, "An efficient probabilistic approach for graph similarity search (technical report)," 2017, http://www.cs.kent.edu/~xlian/TR/techreport.pdf.

[24] Wikipedia. (2017) Chain rule. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Chain_rule_(probability)&oldid=801944140

[25] N. E. Day, "Estimating the components of a mixture of normal distributions," *Biometrika*, vol. 56, no. 3, pp. 463–474, 1969.

[26] Wikipedia, "Continuity correction - wikipedia," 2017. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Continuity_correction

[27] M. H. DeGroot and M. J. Schervish, *Probability and Statistics: Pearson New International Edition*. Pearson Higher Ed, 2013.

[28] K. Riesen and H. Bunke, "Iam graph database repository for graph based pattern recognition and machine learning," *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 287–297, 2008.

[29] H. Jeffreys, "An invariant form for the prior probability in estimation problems," in *Proceedings of the Royal Society of London a: mathematical, physical and engineering sciences*, vol. 186, no. 1007. The Royal Society, 1946, pp. 453–461.

[30] Wikipedia, "Jeffreys prior," 2017. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Jeffreys_prior&oldid=777352107

[31] D. N. Zaharevitz. (2015) Aids antiviral screen data. [Online]. Available: https://wiki.nci.nih.gov/display/NCIDTPdata/AIDS+Antiviral+Screen+Data

[32] Wikipedia. (2017) F1 score. [Online]. Available: https://en.wikipedia.org/w/index.php?title=F1_score&oldid=817504041

[33] X. Zhao, C. Xiao, X. Lin, W. Wang, and Y. Ishikawa, "Efficient processing of graph similarity queries with edit distance constraints," *The VLDB Journal*, vol. 22, no. 6, pp. 727–752, 2013.

[34] X. Zhao, C. Xiao, X. Lin, Q. Liu, and W. Zhang, "A partition-based approach to structure similarity search," *Proceedings of the VLDB Endowment*, vol. 7, no. 3, pp. 169–180, 2013.

[35] Y. Liang and P. Zhao, "Similarity search in graph databases: A multi-layered indexing approach," in *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE, 2017, pp. 783–794.

[36] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[37] K. Riesen, A. Fischer, and H. Bunke, "Improving approximate graph edit distance using genetic algorithms," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition*. Springer, 2014, pp. 63–72.

[38] Wikipedia, "Digamma function," 2017. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Digamma_function&oldid=767319315

APPENDIX A
PROOF OF THEOREM 1

Proof: Since GED satisfies the triangle inequality, we have

$$GED(G_1, G_2) \leq GED(G_1, G_1') + GED(G_1', G_2') + GED(G_2', G_2)$$
$$GED(G_1', G_2') \leq GED(G_1', G_1) + GED(G_1, G_2) + GED(G_2, G_2')$$

According to Definition 1, adding a vertex or an edge with the virtual label ε is not counted as a graph edit operation. Therefore, we have:

$$GED(G_1, G_1') = GED(G_1', G_1) = 0$$
$$GED(G_2, G_2') = GED(G_2', G_2) = 0$$

Therefore, $GED(G_1, G_2) \leq GED(G_1', G_2') \leq GED(G_1, G_2)$.
That is, $GED(G_1, G_2) = GED(G_1', G_2')$. ∎

APPENDIX B
PROOF OF THEOREM 2

Proof: Let the sets of branches rooted at *virtual* vertices in G_1', G_2' be ΔB_{G_1} and ΔB_{G_2}, respectively. We have:

$$|B_{G_1'} \cap B_{G_2'}| = |B_{G_1} \cap B_{G_2}| + |\Delta B_{G_1} \cap B_{G_2}| + |B_{G_1} \cap \Delta B_{G_2}| + |\Delta B_{G_1} \cap \Delta B_{G_2}|$$

Since branches rooted at *virtual* vertices are not isomorphic to any other branches, we also have:

$$|\Delta B_{G_1} \cap B_{G_2}| = |B_{G_1} \cap \Delta B_{G_2}| = |\Delta B_{G_1} \cap \Delta B_{G_2}| = 0$$

Therefore, $|B_{G_1'} \cap B_{G_2'}| = |B_{G_1} \cap B_{G_2}|$.
From the definitions of branches and extended graphs, we have:

$$\max\{|V_1|, |V_2|\} = \max\{|V_1'|, |V_2'|\}$$

From the definition of GBD, we can obtain:

$$GBD(G_1, G_2) = \max\{|V_1|, |V_2|\} - |B_{G_1} \cap B_{G_2}| = \max\{|V_1'|, |V_2'|\} - |B_{G_1'} \cap B_{G_2'}| = GBD(G_1', G_2')$$

That is, $GBD(G_1, G_2) = GBD(G_1', G_2')$. ∎

APPENDIX C
CLOSED FORMS OF EQUATIONS

A.

$$\Lambda_1(G_1', G_2'; \tau, \varphi) = Pr[GBD = \varphi \mid GED = \tau] = \sum_x \Omega_1(x, \tau) \sum_m \Omega_2(m, x, \tau) \sum_r \Omega_3(r, \varphi) \Omega_4(x, r, m) \quad (27)$$

$$\Omega_1(x, \tau) = \mathcal{H}\left(x; |V_1'| + \binom{|V_1'|}{2}, |V_1'|, \tau\right) \quad (28)$$

$$\Omega_2(m, x, \tau) = \binom{\binom{|V_1'|}{2}}{\tau - x}^{-1} \sum_{t=0}^{m} (-1)^{m-t} \binom{|V_1'|}{m} \binom{m}{t} \binom{\binom{t}{2}}{\tau - x} \quad (29)$$

$$\Omega_3(r, \varphi) = \binom{r}{r - \varphi} \cdot \frac{(\mathbb{D}-1)^{\varphi}}{\mathbb{D}^r} \quad (30)$$

$$\Omega_4(x, r, m) = \mathcal{H}\left(x + m - r; |V_1'|, m, x\right) \quad (31)$$

$$\mathcal{H}(x; M, K, N) = \binom{K}{x}\binom{M-K}{N-x}\binom{M}{N}^{-1} \quad (32)$$

$$\mathbb{D} = |\mathcal{L}_V| \cdot \binom{|V_1'| + |\mathcal{L}_E| - 1}{|\mathcal{L}_E|} \quad (33)$$

B.

$$Pr[GED] = \frac{1}{C}\sqrt{\sum_{\varphi=0}^{2\tau} \Lambda_1 \cdot Z^2} \quad (34)$$

$$Z = \frac{1}{\Lambda_1}\left\{\sum_x \Omega_1 \sum_m \frac{\partial \Omega_2}{\partial \tau} \sum_r \Omega_3 \Omega_4 + \sum_x \frac{\partial \Omega_1}{\partial \tau} \sum_m \Omega_2 \sum_r \Omega_3 \Omega_4\right\} \quad (35)$$

where Λ_1 is defined in Equation (27), and Ω_1, Ω_2, Ω_3 and Ω_4 are defined in Equations (28)∼(32), respectively. In addition, we have:

$$\frac{d}{d\tau}\Omega_1(x, \tau) = \binom{\frac{1}{2}\upsilon(\upsilon+1)}{\tau}\binom{\upsilon}{x}\binom{\frac{1}{2}\upsilon(\upsilon-1)}{\tau-x} \cdot F_1 \quad (36)$$

$$\frac{d}{d\tau}\Omega_2(m, x, \tau) = \binom{\frac{1}{2}\upsilon(\upsilon-1)}{\tau-x}^{-1}\binom{\upsilon}{m} \cdot F_2 \cdot \sum_t^m F_3 \cdot F_4 \quad (37)$$

$$F_1 = H(\tau) - H(\tfrac{1}{2}\upsilon(\upsilon+1) - 2\tau) - H(\tau - x) + H(x - \tau + \tfrac{1}{2}\upsilon(\upsilon-1)) \quad (38)$$

$$F_2 = \psi(\tau - x + 1) - \psi(x + 1 - \tau + \tfrac{1}{2}\upsilon(\upsilon-1)) \quad (39)$$

$$F_3 = (-1)^{m-t}\binom{m}{t}\binom{\frac{1}{2}t(t-1)}{\tau-x} \quad (40)$$

$$F_4 = 1 + \psi(x + 1 - \tau + \tfrac{1}{2}t(t-1)) - \psi(\tau - x + 1) \quad (41)$$

Here, υ is short for $|V_1'|$, τ is short for of GED, and x, m, t are summation subscripts in Equation (27). $H(n)$ is the n-th Harmonic Number, and $\psi(\cdot)$ is the Digamma Function [38].