Graph Similarity Search with Edit Distance Constraint in Large Graph Databases

Weiguo Zheng
Peking University
Beijing, China
zhengweiguo@pku.edu.cn

Lei Zou*
Peking University
Beijing, China
zoulei@pku.edu.cn

Xiang Lian
University of Texas - Pan American,
Texas, USA
lianx@utpa.edu

Dong Wang
Peking University
Beijing, China
wangd@pku.edu.cn

Dongyan Zhao
Peking University
Beijing, China
zhaody@pku.edu.cn

ABSTRACT

Due to many real applications of graph databases, it has become increasingly important to retrieve graphs g (in graph database D) that approximately match with query graph q, rather than exact subgraph matches. In this paper, we study the problem of graph similarity search, which retrieves graphs that are similar to a given query graph under the constraint of the minimum edit distance. Specifically, we derive a lower bound, branch-based bound, which can greatly reduce the search space of the graph similarity search. We also propose a tree index structure, namely b-tree, to facilitate effective pruning and efficient query processing. Extensive experiments confirm that our proposed approach outperforms the existing approaches by orders of magnitude, in terms of both pruning power and query response time.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: DATABASE MANAGEMENT—*Database applications*

Keywords

Graph Edit Distance; Lower Bound; Graph Database

1. INTRODUCTION

Recently, graph data models have attracted increasing research interests, because many data types in various applications can be modeled as graphs, such as chemical compounds [2], social networks [16], road networks [1], and Semantic Web [20]. The growing popularity of graph data requires efficient graph data management techniques. Among these, (sub)graph queries (i.e., given a query graph q, finding all graphs g in a graph database D, such that q is (sub)graph isomorphic to g) have been extensively studied [5, 18].

However, the real-life graphs may have many noises, such as protein-protein-interaction networks [10]. In this case, the exact (sub)graph isomorphism may lead to empty results to the query. Therefore, we need to design an *robust* solution to find graphs that are of interest to users even in the presence of noises and errors in the graph database. An interesting topic is to study graph similarity search, which retrieves all graphs g (in graph database D) that approximately match with q under some similarity measure constraint. A number of graph similarity measures have been proposed [4,8,13,17]. Among them, two classical measures (i.e., maximum common subgraphs (MCS) [4] and minimum edit distances (MED) [17]) are proposed based on the classical graph theory. Note that the two measures are inter-related [3]. In this paper, we focus on the *minimum edit distance* (MED). As a widely used structural similarity measure, MED is defined as the minimum operation cost (addition, deletion, and substitution) of transforming one graph q to another graph g (formally defined in Definition 2). Compared with other similarity measures, MED is flexible because it can be used in many applications, such as graph classification and graph clustering [11], objects recognizing in computer vision [7], and molecule comparison in chemistry [9].

In this paper, we study the problem of *graph similarity search* (defined later in Section 2) based on the minimum graph edit distance constraint. Since computing the minimum graph edit distance is a NP-hard problem [17], all existing solutions adopt the filter-and-refine framework to speed up query processing. So far, lots of pruning rules have been proposed. Basically, they can be divided into two categories: the *global filter* and the *n-gram* based filter. The lower bounds of graph edit distance between a query graph q and a data graph g are computed. Then, the data graphs whose lower bounds are larger than τ (τ is a user specified threshold) can be filtered out safely.

1. *global filter.* There are two existing global filters. The first one is to utilize the vertex/edge number difference as the lower bound [17]. The second global filter considers the difference of vertex labels and edge labels to further improve the pruning power [19]. Since these methods do not employ the graph structure, the lower bounds are not tight enough for effective pruning.

2. *n-gram based filter*. The basic idea of these methods is to select some small structures as the *n-grams*. We can derive the lower bound based on the common n-grams of the two graphs. Wang et

*corresponding author: Lei Zou, zoulei@pku.edu.cn

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.
Copyright 2013 ACM 978-1-4503-2263-8/13/10 ...$15.00.
http://dx.doi.org/10.1145/2505515.2505723.

al. [15] propose *k-Adjacent Tree* (*k-AT*) algorithm, which defines a n-gram as a tree consisting of a vertex v and the paths starting at v with length no longer than n. Apparently, a single edit operation may affect many *k-AT* trees, especially when k is larger than 2. The star structure used in [17] is exactly the same as *k-AT* when $k = 1$. Specifically, the star-based lower bound in [17] is $dist_S(g_1, g_2) = \frac{\mu(g_1,g_2)}{max\{4,[max\{\delta(g_1),\delta(g_2)\}]+1\}}$, where $\mu(g_1, g_2)$ is the mapping distance between g_1 and g_2 according to the bipartite graph, $\delta(g_1)$ and $\delta(g_2)$ are the maximum degree in g_1 and g_2, respectively. If g_1 or g_2 has a high-degree vertex, the lower bound will be very small. Similar to *k-AT*, Zhao et al. [19] compute the lower bound by employing the *path-based* n-grams. However, these *path-based* n-grams still share many overlapping structures, if there are some high-degree vertices. Therefore, in such a case, the lower bound of the path-based *n*-grams is not tight.

Generally speaking, the main problem of existing n-gram based pruning methods is that the lower bounds may be not tight enough, since existing n-grams have many overlaps and a single edit operation may affect many n-grams. Considering this limitation, we propose a novel method for edit-distance based graph similarity search problem in this paper. We also propose an index structure to enable effective pruning by the lower bound.

Our lower bound still follows the n-gram approach. However, we use a different n-gram, namely *branch*, which is defined as a structure consisting of one vertex and the edges incident to the vertex[1] (*branch* is formally defined in Section 3). The superiority of *branch* lies in that a single edit operation can affect two branches at most. Therefore, the branch-based lower bound is much tighter than existing n-gram methods.

In order to avoid exhaustively checking all data graphs in D one by one, we build an index structure over graphs in D, namely *b-tree*, where all leaves are data graphs and all non-leaf nodes are the information union of their child nodes. This index benefits the search processing greatly.

To summary, in this paper, we make the following contributions.

- We propose to utilize the *branch* to derive a tight lower bound of minimum graph edit distance.
- In order to reduce the search space, we design an index structure, namely *b-tree* (***b***ranch-based *tree*), to facilitate the query processing.
- Extensive experiments over both real and synthetic graphs confirm the effectiveness and efficiency of our proposed approaches.

2. BACKGROUND

In this section, we first formally define our problem in this section, and then briefly review the existing solutions.

2.1 Problem Definition

For the ease of presentation, we consider simple graphs in this paper. A simple graph is an undirected graph attributed that does not contain self-loops or multi-edges, denoted by g. It can be represented by a 6-tuple $g = (V, E, L_V, L_E, \Sigma_V, \Sigma_E)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, Σ_V and Σ_E are the label sets of V and E, respectively. L_V and L_E are label functions that assign labels to vertices and edges, respectively.

Definition 1. *(Subgraph Isomorphism, Graph Isomorphism). A subgraph isomorphism from g_1 to g_2 is an injection function $f : V(g_1) \rightarrow V(g_2)$ that satisfies 1) $\forall\ v \in V(g_1)$, $f(v) \in V(g_2) \wedge L_V(v) = L_V(f(v))$, and 2) $\forall e(v_1, v_2) \in E(g_1)$, $e(f(v_1), f(v_2)) \in E(g_2) \wedge L_E(e(v_1, v_2)) = L_E(e(f(v_1), f(v_2)))$.*

Let $g_1 \sqsubseteq g_2$ denote that a graph g_1 is subgraph isomorphic to another graph g_2. If $g_1 \sqsubseteq g_2 \wedge g_2 \sqsubseteq g_1$, g_1 and g_2 are graph isomorphic to each other, denoted as $g_1 = g_2$.

There are six primitive edit operations on a graph g [17]: insert an isolated vertex with label, delete an isolated vertex, substitute a vertex label, insert an edge between two vertices, delete an edge, and substitute an edge label. Given two graphs g_1 and g_2, there exists a sequence of primitive edit operations to transform g_1 to g_2, such as, $g_1 = g_1^0 \rightarrow g_1^1 \rightarrow ... \rightarrow g_1^k = g_2$. We may have different operation sequences to transform g_1 to g_2. The minimum graph edit distance (dissimilarity) between two graphs is measured by the shortest operation sequence length, as defined as follows.

Definition 2. *(Minimum Graph Edit Distance). Given two graphs g_1 and g_2, their minimum graph edit distance is defined as the minimum number of primitive operations needed to transform g_1 to g_1', s.t., $g_1' = g_2$, denoted by $ged(g_1, g_2)$.*

Given the definition of minimum graph edit distance (or called graph edit distance if there is no ambiguity in the context), we formalize the problem of this paper as follows.

Problem Statement
(*Graph Similarity Search*) Given a query graph q, a set of data graphs $D = \{g_1, g_2, \cdots, g_{|D|}\}$, and a distance threshold τ, find all graphs g_i in D s.t. $ged(q, g_i) \leq \tau$.

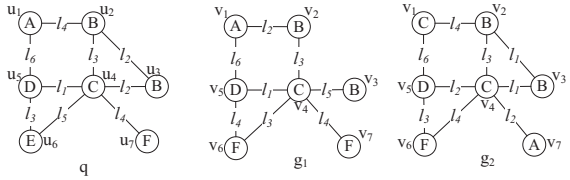


Figure 1: Example of a query graph q and two data graphs g_1 and g_2.

Example 1 below is a graph similarity search running example that is used throughout the paper.

EXAMPLE 1. *Figure 1 shows a query graph q and two data graphs g_1 and g_2. $ged(q, g_1) = 6$, $ged(q, g_2) = 7$. If the edit distance threshold is $\tau = 3$, neither g_1 nor g_2 is the answer, since edit distances of both graphs are larger than 3.*

Since most existing graph similarity search algorithms follow the filter-and-verification framework, it is critical to efficiently estimate the lower bound as tight as possible.

2.2 Existing Solutions

2.2.1 Global Filter

Number Count Filter [17]. The graph edit distance $ged(q, g)$ of two graphs g and q will not be smaller than $dist_N(q, g) = ||V(g)| - |V(q)|| + ||E(g)| - |E(q)||$.

Label Multiset Filter [19]. This lower bound is computed as $dist_M(q, g) = \Gamma(M_V(q), M_V(g)) + \Gamma(M_E(q), M_E(g))$, where $\Gamma(X, Y) = max(|X|, |Y|) - |X \cap Y|$, $M_V(g)$ and $M_E(g)$ are the multisets of vertex and edge labels, respectively.

[1]A branch edge does not include the other endpoint except for the branch center.

2.2.2 n-gram based Filter

C-star. The star structure defined in [17] is a tree of a single level rooted at a vertex. A bipartite graph can be constructed with two star sets $S(q)$ and $S(g)$ of q and g, where each star in $S(q)$ and $S(g)$ is a vertex, and star pair (s_i, s_j) is an edge ($s_i \in S(q)$, $s_j \in S(g)$) weighted with edit distance between s_i and s_j. Assume that the minimum weight matching in the bipartite graph is $\mu(q, g)$. The star-based lower bound of edit distance between graphs q and g can be computed according to Equation 1, where $\delta(q)$ and $\delta(g)$ are the maximum degree in q and g respectively.

$$dist_S(q,g) = \frac{\mu(q,g)}{max\{4, [max\{\delta(q), \delta(g)\}] + 1\}} \quad (1)$$

EXAMPLE 2. *Consider the graphs q, g_1, and g_2 in Figure 1, where $\tau = 3$, $\mu(q, g_1) = 14$, $\mu(q, g_2) = 16$. We have $dist_S(q, g_1) = 14/6$, and $dist_S(q, g_2) = 16/6$. Since $dist_S(q, g_1) < \tau$ and $dist_S(q, g_2) < \tau$ hold, neither g_1 nor g_2 is pruned by this filter.*

Although star-based filter captures some structure information, c-stars may have lots of overlapping structures, which results in a huge penalty by $max\{4, [max\ \{\delta(q),\delta(g)\} + 1]\}$. Obviously, if graph q or g has some large-degree vertices, the lower bound will be very small caused by the large denominator. This problem motivate us to find an effective filter (i.e., less overlapping structures). That is the *branch filter* proposed in Section 3.

Tree-based n-grams (*k-AT* [15]). It defines a n-gram as a tree consisting of a vertex v and the paths starting from v with length no larger than n. Its main principle is based on the observation that if $ged(q, g) \leq \tau$, graphs q and g must share at least

$$dist_T(q,g) = \max(|V(q)| - \tau \cdot D_t, |V(g)| - \tau \cdot D_t(g)) \quad (2)$$

common n-grams, where D_t is the maximum number of tree-based n-grams that can be affected by an edit operation.

EXAMPLE 3. *Consider q, g_1, and g_2 in Figure 1, where τ=1, n=1. $dist_T(q, g_1) = 1$, $dist_T(q, g_2)$= 1. Actually, q has 1 and 0 common tree-based 1-gram with g_1 and g_2, respectively. Hence, g_2 can be pruned, whereas g_1 will pass the filter. If $\tau \geq 2$, both $dist_T(q, g_1)$ and $dist_T(q, g_2)$ will be smaller than 0. Thus, neither g_1 nor g_2 can not be pruned.*

Path-based n-grams [19]. It defines a n-gram as a path of length n. If $ged(q, g) \leq \tau$, graphs q and g must share at least

$$dist_P(q,g) = \max(|MG(q)| - \tau \cdot D_p, |MG(g)| - \tau \cdot D_p(g)) \quad (3)$$

common n-grams, where $MG(q)$ and $MG(g)$ denote the multisets of n-grams in graphs q and g, and D_p is the maximum number of path-based n-grams that can be affected by one edit operation. Clearly, if there is a high-degree vertex, there are many paths containing the vertex. It also means that D_p is very large, which incurs to low pruning power.

EXAMPLE 4. *Consider q, g_1 and g_2 in Figure 1, $\tau = 1$, and $n = 1$. $dist_P(q, g_1)$ = 2, $dist_P(q, g_2)$ = 2. Actually, q has 4 and 2 common path-based 1-grams with g_1 and g_2, respectively. Hence, both g_1 and g_2 can pass this filter. If $\tau \geq 2$, both $dist_P(q, g_1)$ and $dist_P(q, g_2)$ will be smaller than 0. Thus, neither g_1 nor g_2 can be pruned.*

In order to improve the pruning power, we propose a novel filter, namely the *branch*-based filter(Section 3).

3. BRANCH-BASED FILTER

As discussed in the previous section, the pruning abilities of global filters and existing n-gram filters are limited. In order to address these problems, we propose to use the "branch" as the n-gram. It has two benefits. First, branch filters can provide the tighter lower bound than existing ones. Second, by using the branch structures, it is easy to devise an effective index to speed up the query processing. Specifically, we present the branch filter in this section.

Definition 3. *(Branch Structure) A branch structure b is a vertex v and the multiset of edge labels incident to v, represented by $b(v) = (l_v, ES)$, where $l_v = L_V(v)$ is the label of the root vertex, and $ES = \{L_E(e)|$ edge e is adjacent to v $\}$ is the multiset of edge labels adjacent to v.*

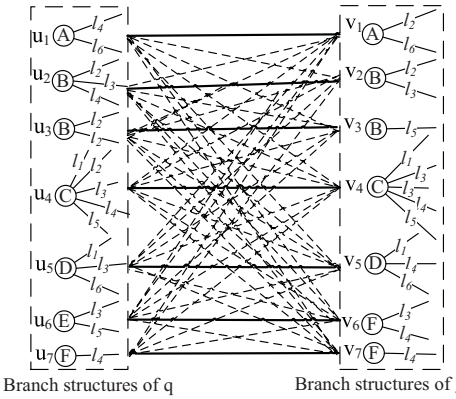


Figure 2: Branch structures of q and g_1.

Figure 2 shows the branches of graphs q and g_1 in Figure 1. Different from stars in [17], a branch only considers edges adjacent to the root vertex. Thus, one edit operation affects two branches at most regardless of the degrees of the vertices. Therefore, the pruning power of branch-based filter is much more stable than other existing n-gram filters, since the pruning ability of existing lower bounds depends on the maximum vertex degree in graphs. According to the definition of branches, we define the distance of two branches as follows.

Definition 4. *(Branch Distance) Given two vertices v_1 and v_2, their branches are denoted as $b_1 = (l_1, ES_1)$ and $b_2 = (l_2, ES_2)$. The branch distance between b_1 and b_2 is defined as follows:*

$$bed(b_1, b_2) = T(l_1, l_2) + \frac{\Gamma(ES_1, ES_2)}{2}$$

where

$$T(l_1, l_2) = \begin{cases} 0, & if \quad l_1 = l_2, \\ 1, & otherwise. \end{cases}$$

$$\Gamma(ES_1, ES_2) = max\{|ES_1|, |ES_2|\} - |ES_1 \cap ES_2|$$

Given two graphs q and g, we can enumerate all branch structures of q and g to obtain two sets, $B(q)$ and $B(g)$, respectively. Thus, we can construct a bipartite graph like that in Figure 2, where vertices represent branches and edges represent transformations between any two branches (from $B(q)$ and $B(g)$ respectively) weighted with the their pairwise *branch edit distance* (defined in Definitin 4).

Definition 5. *Given two multisets of branches $B(q)$ and $B(g)$ with the same cardinality (we can add some branches if $|B(q)| \neq |B(g)|$), and assume P:$B(q) \rightarrow B(g)$ is a bijection. The mapping distance between $B(q)$ and $B(g)$ is*

$$\lambda(q,g) = \min_{P} \sum_{b_i \in B(q)} bed(b_i, P(b_i)) \quad (4)$$

Clearly, the bijection P in Equation 4 is the minimum weighted match in the bipartite graph. Based on the distance between $B(q)$ and $B(g)$ (i.e., $\lambda(q,g)$), we can obtain a lower bound of the edit distance between q and g, as shown in the following theorem.

Theorem 1. *Given two graphs q and g, their graph edit distance and the mapping distance between $B(q)$ and $B(g)$ satisfy the inequality: $ged(q,g) \geq dist_B(q,g) = \lambda(q,g)$, where $B(q)$ and $B(g)$ are the branch structure multisets of q and g respectively, the branch structure-based lower bound is denoted as $dist_B(q,g)$.*

PROOF. Let $P = (p_1, p_2, ..., p_k)$ be an alignment transforming q to g. Accordingly, there is sequence of graph $q = q^0 \rightarrow q^1 \rightarrow ... \rightarrow q^k = g$, where $q^i \rightarrow q^{i+1}$ indicates transforming q^i to q^{i+1} by operation p_i. Assume that there are k_1 edge insertion/deletion/relabeling operations, k_2 vertex insertion/deletion/relabeling operations in P, then $k_1 + k_2 = ged(q,g)$.

1) Edge Insertion/Deletion/Relabeling: If an edge is inserted or deleted or relabeled over the graph q^i, only two branches are affected. Thus we can know that $\lambda(q^i, q^{i+1}) \leq 2/2 = 1$ in the case of inserting or deleting or relabeling an edge over q^i.

2) Vertex Insertion/Deletion/Relabeling: As discussed in [17], a vertex can be deleted only on the condition that it is an isolated vertex, and we can only insert an isolated vertex. If a vertex is inserted or deleted over q^i, $\lambda(q^i, q^{i+1}) = 1$. When the label of a vertex v is relabeled, only the branch rooted at v is affected. Hence, $\lambda(q^i, q^{i+1})$ is 1. Above all, we have the following inequality:

$$\lambda(q,g) \leq 1 \cdot k_1 + 1 \cdot k_2 \leq 1 \cdot (k_1 + k_2) \leq ged(q,g).$$

□

EXAMPLE 5. *Consider graphs q, g_1, and g_2 in Figure 1. According to Theorem 1, $dist_B(q,g_1)$ = 4 and $dist_B(q,g_2)$ = 6, both of which are larger than 3. Hence, g_1 and g_2 can be pruned out safely when $\tau \leq 3$*

As described in Example 5, graphs g_1 and g_2 can be pruned safely employing the branch-based filter. On the contrary, neither g_1 nor g_2 can be filtered out by exiting lower bounds.

4. INDEX AND QUERY PROCESSING

In this section, we first introduce the *b-tree* index, and then give the query processing followed by the construction of *b-tree*.

4.1 b-Tree Index

Given a query q and a database D, we need to exhaustively check the lower bound of $ged(q,g)$ (i.e., branch filter $dist_B(q,g)$) for all graphs g ($\in D$) one by one. Obviously, this is a long and tedious process, especially when $|D|$ is very large. In order to avoid the sequential scan, we propose an index *b-tree* (because it stores the ***b***ranches of each graph $g \in D$) as follows. Generally, *b-tree* is a height-balanced tree, analogue to B^+-tree and R-tree.

Definition 6. *A* b-tree *is a height-balanced tree, where*

(1) Each leaf node stores $B(g)$, the branch set of g, corresponding to the graph g.

(2) Each intermediate node N is union of all its child nodes, i.e., $B(N) = B(N_1) \cup B(N_2) \cup \cdots \cup B(N_m)$, where $N_1, N_2, \cdots, N_m$ are the child nodes of N, "$\cup$" is the union operation.

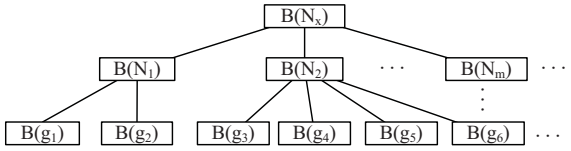


Figure 3: *b-tree* Index.

Figure 3 shows an example of *b-tree* index structure. The construction of *b-tree* is similar to *B-tree* and *R-tree*. We will discuss *b-tree* construction in Section 4.3. Assuming that the *b-tree* has been built, we focus on how to utilize *b-tree* to perform the query.

The general framework is as follows. Given a query graph q, we traverse the *b-tree* starting from the root. Considering an intermediate node N_i (in *b-tree*), we define the directed branch-based distance (denoted as $dist_{DB}(q, N_i)$) between q and N_i in Theorem 1. If this distance is larger than τ, all the descendants of N_i can be pruned safely. The following definition and theorem show the details of the above pruning strategy.

Definition 7. *The* directed branch-based distance *from a query graph q to an intermediate node N_i, denoted as $dist_{DB}(q, N_i)$, is the minimum edit distance of transforming $B(q)$ into $B(q)'$, such that $B(q)' \subseteq B(N_i)$.*

To compute $dist_{DB}(q, N_i)$, we only need to add $|B(q)|$ blank branches (dummy branches without any vertex and edges) into $B(N_i)$, and then construct a bipartite graph according to $B(q)$ and $B(N_i)$. Finally, compute the minimum weighted match in the bipartite graph using Hungarian algorithm [6].

Theorem 2. *If the directed distance $dist_{DB}(q, N_i) \geq (\tau + 1)$, all the children nodes of N_i can be pruned.*

PROOF. Since N_i is the union of its children nodes, the common branches between q and N_i must be more than that between q and N_j, where N_j is one child node of N_i . □

4.2 Query Processing

In this subsection, we propose the query algorithms for graph similarity search. Note that, we only focus on the filtering process, i.e., finding candidates. Any graph edit distance algorithm (such as [19]) can be used in the verification process.

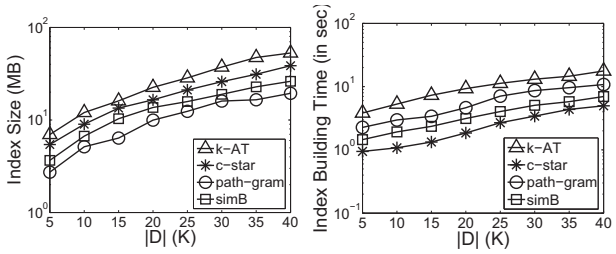
Given a query graph q, we traverse the index *b-tree* starting from the root. For an intermediate node N_i, we compute the directed distance $dist_{DB}(q, N_i)$. If $dist_{DB}(q, N_i) \geq (\tau + 1)$, we can safely prune the subtree rooted at node N_i. Otherwise, the subtree will be accessed. Furthermore, if the current node is a leaf node g, we need to compute the branch lower bound $dist_B(q,g)$ between q and g.

4.3 b-Tree Construction

Since *b-tree* is analogue to *R-tree*, we can build *b-tree* by inserting the graphs sequentially. An insertion operation begins at the root and iteratively chooses a child node until it reaches a leaf node. The given graph is inserted at this leaf node. The main challenge of insertion is the criterion for choosing a child node. We define the similarity between a graph g and a node N_i as the directed branch-based distance between $B(q)$ and $B(N_i)$ (defined in Definition 7). We omit more details about the *b-tree* construction, since it is similar to *R-tree*.

Table 1: Dataset statistics

Dataset	DBSize	Avg |V|	Avg |E|	Avg $\|L_V\|$	Avg $\|L_E\|$	max d
AIDS	42,687	45.7	47.71	4.37	2.06	4
ER	100,000	64.86	157.07	9.39	43.53	7



(a) Index Size (b) Index Construction Time

Figure 4: Offline Performance

5. EXPERIMENTS

In this section, we evaluate the performance of our proposed method (denoted as *simB*), and compare with *c-star* [17], *k-AT* [15] and *path-gram* [19] over both real and synthetic datasets.

5.1 Datasets and Setup

We use real and synthetic datasets in our experiment, described as follows.

Real Dataset. AIDS is an antivirus screen compound dataset from the Developmental Theroapeutics Program in NCI/NIH [2].

Synthetic Dataset. The synthetic graph model is used in our experiments, namely, *E*rdos *R*enyi (denoted as ER). In ER model, N vertices are connected by M randomly chosen edges.

The statistics of the datasets are listed in Table 1, where d is the vertex degree. We randomly select 100 graphs from each dataset as its query graphs, and average the query response time.

In this paper, all experiments are conducted on a P4 3.0GHz machine with 4G RAM running Linux. All programs were implemented in C++. The length of grams in *k-AT* and *path-gram* are set to be 1 and 3, which are the suggested parameter values in [19].

5.2 Evaluating Offline Performance

In this section, we evaluate the offline performance of our method. Due to the space limitation, we only report the index size and index construction time in AIDS dataset, as shown in Figure 4.

Since the depth of tree in is 1 for *k-AT*, it is just the star structures defined in c-star. Because the size of branch is smaller than the size of star, and we assign each branch an unique id to reduce the index space, the space cost of *simB* is competitive in index size. However, *simB* needs to build the *b-tree* index as presented in Section 4. Hence, it is not the most time efficient among these apporaches. Since c-star only needs to enumerate all the star structures in the query graph and data graphs, it is superior to all the other methods in terms of the index construction time.

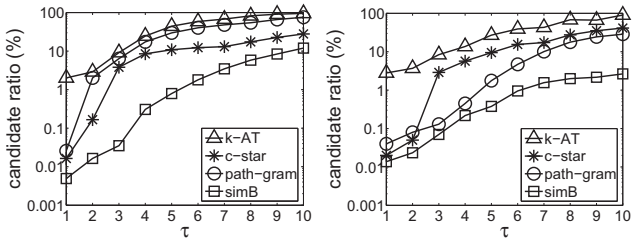
5.3 Evaluating Online Performance

5.3.1 *Candidate Ratio* $vs.$ τ

In order to evaluate the pruning ability of these approaches, we propose a metric, namely candidate ratio (denoted as *canRatio*), which is defined in Equation 5.

$$canRatio = \frac{candidate\ size}{|D|} \quad (5)$$

[2]http://dtp.nci.nih.govdocsaidsaids_data.html



(a) Results on AIDS (b) Results on ER(100K)

Figure 5: $canRatio\ vs.\ \tau$ **in Graph Similarity Search**

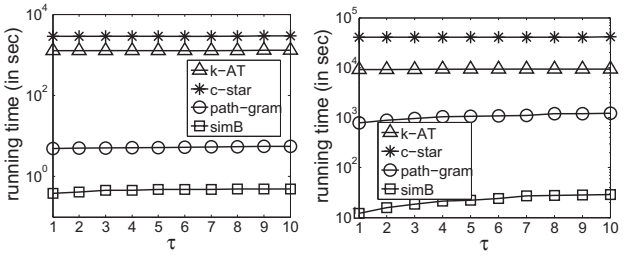
where $|D|$ is the number of graphs in the database.

In this subsection, we fix the datasets and vary the threshold τ from 1 to 10. Figure 5 presents the *canRatio*s of different methods in AIDS and ER(100k), where the y-axis represents the candidate ratio generated by these methods.

According to the definition of *canRatio*, it is clear that the smaller the *canRatio* is, the higher the prune ability of filters is. As shown in Figures 5(a) and 5(b), the candidate ratios generated by all these methods increase with the increasing of τ. It is because that larger threshold will produce more candidates. Note that, the candidate ratio of our proposed method *simB* is the lowest, i.e., it has the strongest prune power. Thus, it confirms that our method is more effective compared with *c-star* [17], *k-AT* [15] and *path-gram* [19].

5.3.2 *Filtering Time* $vs.$ τ

In this subsection, we evaluate the query efficiency of these approaches. Since c-star needs to construct the bipartite graph between two sets of star structures and it does not employ any index structure, it is the most inefficient compared other methods. On the contrary, the branch distance is easy to compute. Moreover, we carefully devise the *b-tree* index which benefits the query processing. Hence, the query response time of our method *simB* is much less than the other three filters by orders of magnitude as shown in Figure 6.

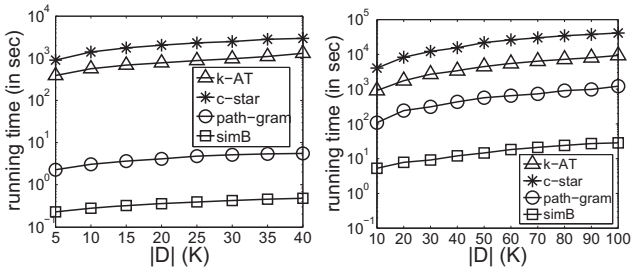


(a) Results on AIDS (b) Results on ER(100K)

Figure 6: Running Time $vs.$ τ **in Graph Similarity Search**

5.3.3 *Filtering Time* $vs.$ $|D|$

We also study the scalability of *simB* together with other methods as shown in Figure 7. We fix the threshold τ to be 3, and vary the the size of datasets. We randomly select some subsets from AIDS(full) and ER(100K). As shown in Figure 7, the time consumed by all these approaches are almost linear of the size of datasets, because the prune ability is stable in datasets of different size. What is more, our method *simB* is faster than the other three filters by orders of magnitude.



(a) Results on AIDS (b) Results on ER(100K)

Figure 7: Running Time $vs.$ $|D|$ in Graph Similarity Search

6. RELATED WORK

In many real world applications, various noisy and incomplete data drive the research of approximate (sub)graph search. Hence, performing an exact graph matching probably returns none match or incorrect ones. It is crucial to find the approximate matches. SAGA [12] proposes a distance model for computing graph similarity, which permits node gaps, node mismatches, and graph structural differences. G-Ray [14] defines the similarity based on model of random walk with restart. TALE [13] introduces the neighborhood index (NH-Index), the size of which is linear in the number of nodes in the database. Ness [8] proposes a graph similarity measure based on neighborhood information, under this measure subgraph similarity search is NP hard.

All above approaches define some "heuristic" distances to model the similarity between two graphs. They do not discuss the relationships between the heuristic functions and some classical graph similarity distances. In this paper, we focus on graph edit distance. It is based on well-found graph theory. Recently, graph edit distance similarity (sub)graph search has attracted extensive attentions [15,17,19]. Since computing graph edit distance is also a NP-hard problem, all existing methods also adopt the filter-and-verification framework. Lots of lower bounds are proposed to perform the pruning. We have reviewed these approaches in Section 2.2. The key problem is that the pruning power of existing solutions depends on the maximum vertex degree in graphs. Extensive experiments show that our proposed lower bounds are much better than existing ones.

7. CONCLUSIONS

Considering the limitations of existing approaches, we present a novel method for edit-distance based similarity graph similarity search problem. An effective lower bound based on the branch structures is first proposed. To facilitate the query processing, we carefully devise a tree index, namely *b-tree*. Extensive experiments over both real and synthetic datasets confirm that our proposed method outperforms the existing approaches significantly.

8. ACKNOWLEDGMENTS

This work was supported by NSFC under Grant No.61003009, 61272344, 61370055. Dongyan Zhao's work was also supported by National High Technology Research and Development Program of China under Grant No. 2012AA011101. Lei Zou's work was partially supported by State Key Laboratory of Software Engineering(SKLSE), Wuhan University, China and CCF-Tencent Open Research Fund.

9. REFERENCES

[1] J. E. Beasley and N. Christofides. Theory and methodology: Vehicle routing with a sparse feasibility graph. *European Journal of Operational Research*, 98(3), 1997.

[2] D. Bonchev and D. H. Rouvray. *Chemical Graph Theory: Introduction and Fundamentals*. Abacus Press, New York, 1991.

[3] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[4] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3-4):255–259, 1998.

[5] H. He and A. K. Singh. Closure-tree: An index structure for graph queries. In *ICDE*, page 38, 2006.

[6] H.W.Kuhn. The hungarian method for the assignment problem. In *Naval Research Logistics*, pages 83–97, 1955.

[7] D. Justice and A. O. Hero. A binary linear programming formulation of the graph edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(8):1200–1214, 2006.

[8] A. Khan, N. Li, X. Yan, Z. Guan, S. Chakraborty, and S. Tao. Neighborhood based fast graph search in large networks. In *SIGMOD Conference*, pages 901–912, 2011.

[9] R. M. Marin, N. F. Aguirre, and E. E. Daza. Graph theoretical similarity approach to compare molecular electrostatic potentials. *J. of Chem. Inf. and Model.*, pages 933–944, 2008.

[10] N. Przulj. Geometric local structure in biological networks. In *IEEE Information Theory Workshop*, pages 131–140, 2007.

[11] A. Robles-Kelly and E. R. Hancock. Graph edit distance from spectral seriation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):365–378, 2005.

[12] Y. Tian, R. C. McEachin, C. Santos, D. J. States, and J. M. Patel. Saga: a subgraph matching tool for biological graphs. *Bioinformatics*, 23(2):232–239, 2007.

[13] Y. Tian and J. M. Patel. Tale: A tool for approximate large graph matching. In *ICDE*, pages 963–972, 2008.

[14] H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In *KDD*, pages 737–746, 2007.

[15] G. Wang, B. Wang, X. Yang, and G. Yu. Efficiently indexing large sparse graphs for similarity search. *IEEE Trans. Knowl. Data Eng.*, 24(3):440–451, 2012.

[16] D. J. Watts, P. S. Dodds, and M. E. J. Newman. Identity and search in social networks. *Science*, 296(5571), 2002.

[17] Z. Zeng, A. K. H. Tung, J. Wang, J. Feng, and L. Zhou. Comparing stars: On approximating graph edit distance. *PVLDB*, 2(1):25–36, 2009.

[18] P. Zhao and J. Han. On graph query optimization in large networks. *PVLDB*, 3(1):340–351, 2010.

[19] X. Zhao, C. Xiao, X. Lin, and W. Wang. Efficient graph similarity joins with edit distance constraints. In *ICDE*, pages 834–845, 2012.

[20] L. Zou, J. Mo, L. Chen, M. T. Özsu, and D. Zhao. gstore: Answering sparql queries via subgraph matching. *PVLDB*, 4(8):482–493, 2011.