计算机科学
COMPUTER SCIENCE

# 高效低索引的图相似性搜索算法

邱珍，郑朝晖

引用本文

邱珍，郑朝晖. 高效低索引的图相似性搜索算法[J]. 计算机科学，2023，50(9)：130-138.

QIU Zhen，ZHENG Zhaohui. Graph Similarity Search with High Efficiency and Low Index [J]. Computer Science，2023，50(9)：130-138.

---

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220700105

# 高效低索引的图相似性搜索算法

邱　珍　　郑朝晖
苏州大学计算机科学与技术学院　江苏 苏州 215006
江苏省网络空间安全工程实验室　江苏 苏州 215006
(20205227108@stu.suda.edu.cn)

**摘　要**　图相似性搜索是在给定的度量标准下查找与查询图相似的图集合,目前大多采用“过滤-验证”的计算框架。针对现有方法中过滤下界不紧密和索引空间占用较大等问题,提出了一种基于查询图分区的多层级过滤、低索引空间占用的图相似性搜索算法 Z-Index。该算法首先通过全局粗粒度过滤得到预候选集;然后提出基于扩展概率的查询图分区算法,并采用层级过滤机制进一步精简候选集,增强下界紧密性;最后引入序列相似性差值计算序列中数据分布的稀疏度,提出分区压缩和差值压缩两种编码压缩算法,并据此构建“零”索引结构,降低索引空间开销。实验结果表明,Z-Index 算法所得下界更加紧密,产生的候选集大小可减少 50%左右,算法执行时间大大缩短,且该算法在索引空间占用极小的情况下仍具有可扩展性。

**关键词**:图相似性搜索;层级过滤;扩展概率;编码压缩;查询图分区
**中图法分类号**　TP391

## Graph Similarity Search with High Efficiency and Low Index

QIU Zhen and ZHENG Zhaohui
School of Computer Science and Technology,Soochow University,Suzhou,Jiangsu 215006,China
Jiangsu Province Cyberspace Security Engineering Laboratory,Suzhou,Jiangsu 215006,China

**Abstract**　Graph similarity search is to search the graph set that is similar to query graph under a measurement,which adopts the “filtering-verification” framework. Aiming at the problems of the existing methods,such as the untight lower bound and the large index space,an improved graph similarity search algorithm(Z-Index) based on query graph partition with multi-level filtering and low index space is proposed. Firstly,the pre-candidate set is obtained by global coarse-grained filtering. Secondly,a query graph partitioning algorithm based on extension probability is proposed,and a hierarchical filtering mechanism is adopted to further shrink the candidate set,so as to enhance the tightness of the lower bound. Finally,the sequence similarity difference is introduced to compute the sparsity of the data contribution. Then partition compression and difference compression algorithm are proposed to construct “zero” index structure,so as to reduce the index space. Experimental results show that Z-Index algorithm has a tighter lower bound,and the candidate set size of Z-Index can be reduced about 50%. Moreover,the algorithm execution time is greatly reduced,and it still shows great scalability in the case of tiny index space.

**Keywords**　Graph similarity search,Hierarchical filtering,Extension probability,Coding compression,Query graph partitioning

## 1　引言

近年来,随着互联网技术的飞速发展,数据量呈指数级增长,实现数据的高效存储与检索至关重要。在大数据时代,由于数据实体具有各自的特征属性且大量数据之间存在相互关联的复杂关系,因此通常将这些数据实体以及数据之间的关系抽象为图结构。面对大规模图数据集,图相似性搜索算法在数据分析中具有重要意义,且已被广泛应用于各个领域,如生化信息学、计算机视觉、模式识别和数据检索等[1-3]。

在图数据集中,对于给定的查询图 $q$ 和编辑距离阈值 $\tau$,根据指定的图相似性度量标准检索所有编辑距离不超过 $\tau$ 的图 $g$ 的过程被称为图相似性搜索。目前,评估图相似性的度量标准有图编辑距离[4]、最大公共子图[5]和图对齐[6]等。其中,图编辑距离(Graph Edit Distance,GED)作为最常用的度量指标,几乎可以评估所有类型的图,精确计算图之间的结构差异。由于图编辑距离计算是 NP-Hard 问题,因此现有方法大多采用“过滤-验证”的思路求解图相似性搜索问题,其性能主要取决于候选集大小、过滤得到候选集的代价,以及图编辑距离的计算开销。在过滤阶段,通常采用索引构建算法和上下界剪枝策略来快速过滤不满足阈值约束的数据图,得到

到稿日期:2022-07-11　返修日期:2022-11-11
基金项目:江苏省高校自然科学研究项目(19KJA550002)
This work was supported by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China(19KJA550002).
通信作者:郑朝晖(zhengzh@suda.edu.cn)

候选集。但过于松弛的过滤下界会导致候选集过大，设计较优的索引结构能缓解这一问题，但会导致索引空间占用较多，然而大部分研究没有考虑到这一性能瓶颈。在验证阶段，要分别精确计算查询图与候选集中数据图的图编辑距离，该过程需要较大的计算开销。如果过滤阶段能够得到精简的候选集，则会大大降低验证阶段的时间消耗。因此，设计高效的过滤机制是优化图相似性搜索算法的重要一环，本文会对该过程做进一步的优化。

针对上述候选集较大和索引空间占用较大等问题，本文对过滤策略做出改进并优化索引空间，提出了一种基于查询图分区的多层级过滤、低索引空间占用的图相似性搜索算法Z-Index，并在不同数据集上进行实验验证。实验结果表明，本文算法能够在低索引空间占用下实现高效查询。

主要工作总结如下：

1）提出了一种基于扩展概率的查询图分区算法，为每个分区引入一个扩展概率值，即顶点或边被分配到当前分区的可能性，将复杂的结构分区过程转换为简单的数值比较，根据该值可以更精确地判断一个分区与数据图是否匹配，提高了过滤效果。

2）提出了层级过滤机制以减少候选集大小。为避免不必要的分区匹配与索引构建，在对查询图分区之前首先采用粗粒度过滤得到预候选集，然后在分区过程中基于子图匹配方法进行过滤以进一步精简候选集，解决了候选集过大的问题。

3）不同于其他研究者对数据库中的图进行分区，本文对查询图分区并建立索引，为每个索引序列引入元素相似性差值，来表征该序列的数据分布稀疏度，并在此基础上提出分区压缩和差值压缩两种编码压缩算法，进而建立“零”索引结构，在降低索引空间的同时大大加快了过滤速度，缓解了在海量数据图中构建索引带来的空间压力。

## 2　相关工作与问题定义

### 2.1　相关工作

针对图相似性搜索问题，国内外学者开展了诸多广泛的研究[7-9]。在验证阶段，常用的图编辑距离算法[10]有：A*GED[11]，DF_GED[12]，D-DF[13]，CSI_GED[14]，Astar-LSa[15]和BSS_GED[16]等。在过滤阶段，Wang等[17]提出了基于树的$q$-gram和基于路径的$q$-gram，通过构建$k$-AT树过滤筛选不符合下界条件的数据图，将$k$-AT索引组织为倒排索引以避免缓慢的顺序搜索，不足之处在于该方法只适用于稀疏图。此后，Zheng等[1]提出了分支距离的概念，设计了新的下界过滤机制，但其算法的时间和空间复杂度较高。在此基础上，Zheng等[18]又提出了基于3种过滤边界的混合过滤方法。上述过滤方法都采用了固定划分子结构的思想，由于子结构之间存在重叠部分，一次图编辑操作可能会影响多个子结构，因此该方法的过滤效果有待优化。针对该问题，Zhao等[19]首次提出了不相交图划分[20]的思想Pars，将数据图划分为$\tau+1$个非重叠子结构，通过子图同构计算过滤数据图，其缺点在于需要较长的索引构造时间和子图同构计算时间，而且随机分区方法会对识别假阳性图造成干扰。此后，Liang等[21]提出了参数化的下界和选择性图划分方法ML-Partition。该方法可以识别更多的假阳性图，减少候选集大小，而对于大量数据图而言，图划分、子图同构计算和倒排索引的构建均需占据大量的时间与空间。

虽然目前已有各种优化的图相似性搜索算法，但是部分改进后的算法仍存在下界不紧密的问题，过滤比例仍然较低，且大多数研究未考虑索引空间占用较大引起的空间消耗问题。基于此，本文提出了一种高效低索引的图相似性搜索算法，在获得较小候选集的同时，能够保证索引占用较低。

### 2.2　问题定义

本文中，将带标签图集合$G$定义为一个三元组：$G=\{V,E,L\}$。其中$V$表示图$G$中的顶点集合，$E\subseteq V\times V$表示边集合，$L$表示标签标记函数。对于一个数据图$g\in G$，用$V_g$和$E_g$分别表示图$g$中的顶点和边集合，用$|V_g|$和$|E_g|$分别表示图$g$中的顶点和边的数量，$|G|$表示图集合$G$中数据图的数量。

**定义1**（图编辑距离[22]，Graph Edit Distance，GED）　数据图$g$转换为查询图$g'$所需的最少编辑距离操作数，用来衡量两个图之间的结构差异。本文使用$GED(g,g')$表示图$g$和$g'$之间的编辑距离。其中图编辑距离操作包括以下6点：

1）插入一个新的孤立顶点$u$；

2）在已有顶点$u$和$v$之间插入新边$e$，$e=(u,v)$；

3）删除一个孤立顶点$u$；

4）删除连接顶点$u$和$v$的边$e$，$e=(u,v)$；

5）修改顶点$v$的标签；

6）修改边$e$的标签。

例1　如图1所示，给定两个图$g_1$和$g_2$，则$GED(g_1,g_2)=5$。其中，$g_1$转换为$g_2$的编辑操作步骤具体体现为：

1）删除连接顶点$C,D$的边；

2）删除连接顶点$A,F$的边；

3）删除连接顶点$C,F$的边；

4）删除顶点$F$；

5）将顶点$C$修改为$E$。

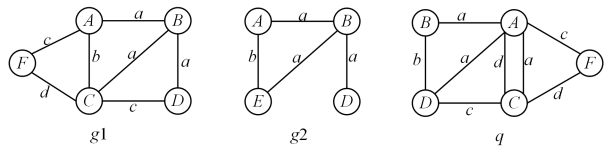



图1　查询图$q$和数据图$g_1,g_2$

Fig. 1　Query graph $q$ and data graphs $g_1,g_2$

**定义2**（不相交图分区）　将数据图根据特定规则划分成两两互斥的独立子结构。

在本文Z-Index算法中，对于一个给定的图$g$，将满足以下条件的分区结果表示为$P(g)=\{p_1,p_2,\cdots,p_i\}$。

1）$\forall i,p_i\subseteq P(g)$；

2）$\forall i,j$且$i\neq j,V_{p_i}\cap V_{p_j}=\varnothing$；

3）$V=\bigcup_{i=1}^{n}V_{p_i}$，$E=\bigcup_{i=1}^{n}E_{p_i}$。

在Z-Index算法的分区过程中，为解决固定分区导致的下界松散问题，提升过滤性能，本文提出了基于扩展概率的查询图分区算法，通过引入扩展概率值，动态计算图中各顶点与边的匹配情况，最终得到满足上述不相交条件

的图分区集合 $P(q)$。

例 2 如图 2 所示，查询图 $q$ 被分为 4 个分区，即 $P(q)=\{p_1,p_2,p_3,p_4\}$。其中任意两个分区都不存在重叠部分，且所有分区的并集为完整的图 $q$。

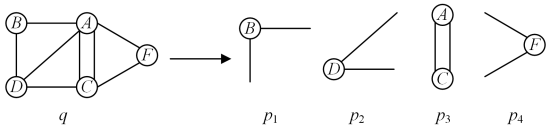



图 2 查询图 $q$ 的分区情况

Fig. 2 Partitions of query graph $q$

**定义 3**(图相似性搜索) 图相似性搜索算法指从一个给定的数据图集合 $G$ 中查找与某一查询图 $q$ 的编辑距离小于或等于阈值 $\tau$ 的所有数据图 $g$ 的集合 $R$。定义如下：

$$R=\{g|GED(g,q)\leqslant\tau\} \tag{1}$$

例 3 如图 1 所示，给定图数据集 $G=\{g_1,g_2\}$ 和查询图 $q$，设编辑距离阈值 $\tau=4$，分别求得 $GED(g_1,q)=3<\tau$，$GED(g_2,q)=7>\tau$，其中满足编辑距离阈值的数据图为 $g_1$，则 $R=\{g_1\}$。

**定义 4**(过滤下界) 对于图 $g$ 和 $q$，本文定义编辑距离下界 $LB(g,q)$ 来实现数据图的过滤，如果 $GED(g,q)\geqslant LB(g,q)>\tau$，即过滤下界大于编辑距离阈值 $\tau$，则图 $g$ 可被过滤，不用精确计算图编辑距离。因此，过滤下界越紧密，越接近真实 $GED$，所得候选集就越小，算法性能越好。

## 3 算法设计与分析

本文 Z-Index 算法主要包括以下 3 个重要组成部分：1)首先基于扩展概率对查询图做分区处理，用查询子图去匹配数据库中的图；2)采用层级过滤机制精简候选集，以此减少验证阶段中图编辑距离的计算次数；3)最后基于编码压缩算法构建“零”索引，降低索引空间占用，使得能在有限的空间内实现高效查询。

本文 Z-Index 算法流程图如图 3 所示，其中，$l$ 表示预设的序列压缩阈值，$gap$ 表示任一序列的元素相似性差值。

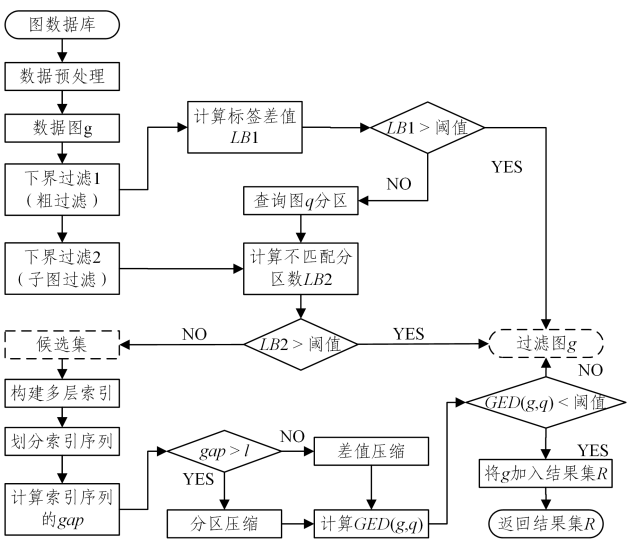



图 3 Z-Index 算法流程图

Fig. 3 Flow chart of Z-Index algorithm

### 3.1 基于扩展概率的查询图分区算法

基于固定大小子结构的图分区方法存在以下缺点：1)忽略了图的拓扑结构信息，随着数据规模的增大，其可扩展性存在局限性；2)固定大小子结构存在大量结构冗余，一次图编辑距离操作可能会影响多个子结构，导致下界过于松散。为增强下界紧密性，提升过滤性能，Z-Index 算法针对分区过程中各区域的匹配情况，提出了基于扩展概率的查询图分区算法。

不同于 Pars[19] 对图数据库中所有数据图进行随机分区进而造成分区冗余，本文对查询图集合根据定义 2 进行不相交图划分，通过引入扩展概率值的概念，将查询图划分为 $\tau+k$ 个非重叠子区域，其中 $\tau$ 为编辑距离阈值，$k$ 为下界参数值。对于一个分区 $p_i$ 而言，其扩展依据为分区大小和分区中顶点与边标签出现的频率。其中，分区大小表示该分区中顶点和边的总数量，即 $|V_{P_i}|+|E_{P_i}|$；顶点标签频率表示该分区所有顶点中每一类顶点出现的次数，即 $\sum_{v\in V_{P_i}} f(L_v)$；同理，边标签频率表示为 $\sum_{e\in E_{P_i}} f(L_e)$。如果查询图中某一分区较大，那么它就越有可能被编辑距离操作所影响，越不容易被匹配到。同理，分区 $p_i$ 中顶点和边标签频率越高，那么它在数据图中出现的概率也越大，越容易被匹配到。因此，根据扩展概率值 $s(p_i)$ 可以快速判断分区 $p_i$ 与图 $g$ 是否匹配，大大提高过滤效果。对分区 $p_i$ 而言，其扩展概率值 $s(p_i)$ 的定义如下：

$$s(p_i)=\frac{\sum\limits_{v\in E_{P_i}}\frac{f(L_{p_i}(v))}{|V_{p_i}|}+\sum\limits_{e\in E_{P_i}}\frac{f(L_{p_i}(e))}{|E_{p_i}|}}{|V_{p_i}|+|E_{p_i}|} \tag{2}$$

其中，$f(L_{p_i}(v))$ 表示分区 $p_i$ 中顶点标签为 $v$ 的顶点数量；$f(L_{p_i}(e))$ 同理。$s(p_i)$ 值越大，表明分区 $p_i$ 越容易被匹配到，即图 $g$ 与查询图 $q$ 相似的概率越大。对顶点 $v$ 而言，其加入到分区 $p_i$ 中的贡献值 $\Delta p_i$ 的定义为：

$$\Delta p_i=|s(p_i\cup\{v\})-s(p_i)| \tag{3}$$

Z-Index 中基于扩展概率的查询图分区算法有两个主要步骤：顶点分配和跨区域边的分配。给定查询图 $q$，则基于扩展概率的查询图分区过程如下：

1)随机选取 $\tau+k$ 个初始顶点，随后这些顶点将被扩展为 $\tau+k$ 个分区。

2) 尝试将初始顶点的邻居顶点加入到每个分区中，根据式(3)计算其对分区 $p_i$ 的贡献值 $\Delta p_i$，将顶点加入到 $\Delta p_i$ 最大的分区中。如果 $\Delta p_i$ 相等，则将顶点随机加入较小的分区中。

3) 重复步骤 2)过程，计算每个区域的邻居顶点对各区域的贡献值 $\Delta p_i$，直到分配完所有顶点。至此，顶点分配结束。

4)跨区域的边分配：所有顶点分配完之后，尝试将边分配到其顶点所在的分区中并计算 $\Delta p_i$，最终将其分配到 $\Delta p_i$ 最大的区域中。

例 4 以图 4 为例，设 $\tau=2$，$k=1$，选择顶点 $B$，$D$，$F$ 作为初始顶点，分别标记所在分区为 $p_1$，$p_2$ 和 $p_3$，计算得 $s(p_1)=s(p_2)=s(p_3)=1$。

首先进行顶点分配，与这些分区相邻且未分配的顶点有 $A$ 和 $B$，由于 $A$ 和 $B$ 以及 $F$ 相邻，因此尝试将顶点 $A$ 分配到 $p_1$ 和 $p_3$ 中，计算得 $\Delta p_1=0.5$，$\Delta p_3=0.5$，则将顶点 $A$ 随机加入分区 $p_1$。由于顶点 $B$ 与 $A$，$D$，$F$ 相邻，因此 $B$ 可被分配

到区域 $p_1$，$p_2$ 和 $p_3$ 中，计算得 $\Delta p_1=0.03$，$\Delta p_2=0.17$，$\Delta p_3=0.17$，则将顶点 $B$ 加入分区 $p_2$ 中。此时未分配顶点集合为 $\{A\}$。由于顶点 $A$ 只与区域 $p_2$ 相邻，因此将 $A$ 加入区域 $p_2$。此时 $p_1=\{B,A,a(B,A)\}$，$p_3=\{F\}$，$p_2=\{D,B,A,b(D,B),d(B,A)\}$，顶点分配结束。

然后进行跨区域的边的分配，跨区域的边集合为 $\{a(B,A),b(A,B),c(A,F),d(B,F)\}$。对于边 $a(B,A)$，计算 $\Delta p_1=0.75-0.67=0.08$，$\Delta p_1=0.4-0.33=0.07$，因为 $\Delta p_1>\Delta p_2$，所以将边 $a(B,A)$ 加入到 $p_1$ 分区中。同理，将其他边加入到对应分区中。最终得到分区信息为：$p_1=\{B,A,a(B,A),a(B,D),b(A,B)\}$ $p_2=\{D,B,A,b(D,B),d(B,A)\}$ $p_3=\{F,d(F,B),c(F,A)\}$。

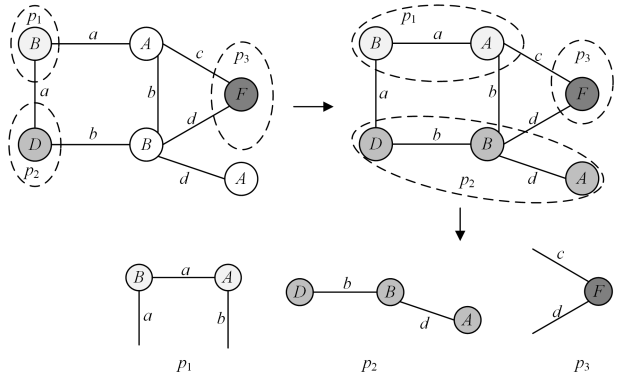



图 4　基于扩展概率的图分区过程

Fig. 4　Graph partitioning based on extension probability

### 3.2　层级过滤机制

为减少验证阶段图编辑距离的计算次数，防止单一过滤造成的数据适应性，Z-Index 算法对过滤过程进行了优化，基于全局粗过滤和子图匹配方法，提出层级过滤机制，精简候选集大小。详细算法过程如下。

1）一级过滤：在分区之前进行粗粒度过滤，计算图 $g\in G$ 与查询图 $q$ 之间的顶点和边数的差值 $LB_1$。定义如下：

$$LB_1(g,q)=\|E_g|-|E_q\|+\|V_g|-|V_q\| \tag{4}$$

其中，$|V_g|$ 和 $|E_g|$ 分别表示图 $g$ 中顶点和边的数量。若 $LB_1(g,q)>\tau$，那么至少需要 $\tau+1$ 次顶点/边的删除或者添加操作才能将 $g$ 转换到 $q$，则 $GED(g,q)\geqslant LB_1(g,q)>\tau$。图 $g$ 与查询图 $q$ 的图编辑距离一定大于阈值 $\tau$，所以在未分区之前就可将图 $g$ 过滤掉，得到预候选集并进行扩展分区，可避免不必要的图分区判断过程。

2）二级过滤：在对预候选集的分区过程中计算不匹配分区数，判断图 $g$ 是否可以被过滤。在对查询图分区的过程中，计算查询图 $q$ 与每个数据图 $g$ 之间的不匹配分区数，记为 $LB_2(g,q)$。如果图 $g$ 中不匹配分区数大于编辑距离阈值 $\tau$，即 $LB_2(g,q)>\tau$，那么该图一定不在编辑距离约束范围内，可以被安全过滤。根据鸽巢原理[23]，每个不匹配分区至少需要一次编辑距离操作才能达到匹配状态，若不匹配分区数大于 $\tau$，则至少需要 $\tau+1$ 次操作，因此不再满足编辑距离阈值条件的约束。如图 2 所示，$p_1\subseteq g_1$，$p_4\subseteq g_1$。$p_1$ 和 $p_4$ 为图 1 中 $g_1$ 的匹配分区，而 $p_2$ 和 $p_3$ 为不匹配分区，所以不匹配分区数为 $2\leqslant\tau$，图 $g_1$ 可能是查询图 $q$ 相似度搜索的结果，可以放入候选集中，进行下一步图编辑距离 GED 验证。

综上所述，经过层级过滤机制后，满足 $LB_1(g,q)>\tau$ 或者 $LB_2(g,q)>\tau$ 的数据图将会被过滤，可以得到更为精简的候选集，大大减少了验证阶段图编辑距离的计算次数。

### 3.3　“零”索引的图相似性搜索算法

不同于其他研究者对数据库中的所有图分区，本文对查询图分区，可以在一定程度上减少索引空间的占用。但是当查询图集合逐渐增大时，时间开销与索引所需空间也会随之增加。本文在 ML-Partition[21] 算法多层索引的基础上，基于编码压缩算法[24-26] 构建“零”索引结构 $ZIndex$，该过程包括两个步骤：建立索引和索引压缩。其中，在索引压缩阶段，本文提出分区压缩和差值压缩两种改进的索引压缩算法，通过计算索引序列的元素相似性差值推断序列中数据分布的稀疏度，进而选出合适的压缩算法，使得在较小的内存下实现高效查询。

1）建立索引

本文对查询图建立 $L$ 层“零”索引结构，具体流程如下：在第 $i(1\leqslant i\leqslant L)$ 层中，基于查询图扩展概率分区算法将查询图 $q$ 划分为 $\tau+k$ 个分区，经过层级过滤机制得到该层对应的候选集 $C_i$，最终候选集合 $C_q=\bigcap_{i=1}^{L}C_i$。对查询图 $q$ 的每一个分区 $p$，维护一个倒排索引表 $I(p)$，保存包含该分区的所有数据图 $g$。由此，可以在数据集 $G$ 中快速查找到含有子图 $p$ 的所有图 $g$。Pars 算法在判断图 $g$ 是否匹配分区 $p$ 时，需要频繁进行子图同构计算。为避免复杂的子图同构计算，Z-Index 在查询图分区的过程中记录图 $g$ 与分区 $p$ 的顶点与边标签的频率，记为 $N(g)$ 和 $N(p)$。如果 $p$ 是图 $g$ 的匹配分区，则表示为 $N(g)\leqslant N(p)$，否则视为不匹配分区。根据以上阐述，将第 $i$ 层中的“零”索引结构定义为：$Z_i^q=(I(p_i),N(p_i))$。

2）索引压缩

本文基于编码算法，提出了两种有效压缩存储 $Z_i^q$ 的方法，构建“零”索引序列 $ZIndex$。首先计算索引序列 $s$ 中的元素相似性差值 $s_{\text{gap}}$，并与序列压缩阈值 $l$ 进行比较。若 $s_{\text{gap}}$ 大于压缩阈值 $l$，则采用分区压缩算法，否则采用差值压缩算法。其中，$s_{\text{gap}}$ 的定义如式(5)所示：

$$s_{\text{gap}}=\frac{\sum|s_{i+1}-s_i|}{|s|},0\leqslant i<|s|-1 \tag{5}$$

其中，$|s|$ 表示序列的 $s$ 的长度。

(1)分区压缩算法

对数据分布不均匀的索引序列而言，统一压缩会降低压缩效果。为解决这一问题，本文提出了基于序列划分的索引分区压缩算法，根据数据分布的稀疏情况选择划分长度 $d$，将其划分为若干 $|s|/d$ 个子序列分别压缩。例如，对于给定的序列 $s=\{1,2,3,4,5,125,130,137,144,158\}$，如果设置 $d=5$，则序列 $s$ 可划分为子序列 $s_1$ 和 $s_2$，其中 $s_1=\{1,2,3,4,5\}$，$s_2=\{125,130,137,144,158\}$，则问题转换为对子序列 $s_1$ 和 $s_2$ 做压缩处理。

(2)差值压缩算法

该算法保持每个划分中的第一个元素不变，然后依次计算此后相邻两个元素的差值，即 $s_1\rightarrow s_1'=\{1,1,1,1,1\}$，$s_2\rightarrow s'_2=\{125,5,7,7,14\}$。最终，对处理过的序列 $s_1'$ 和 $s_2'$ 使用

编码算法压缩。

常用的编码算法[24]有一元编码、哥伦布编码和指数哥伦布编码等。使用不同的编码算法 compress(·)将会得到不同的“零”索引序列 $ZIndex$：

$$ZIndex=\sum_{q\in Q}\sum_{i=1}^{L}ZIndex_i^q \tag{6}$$

其中，$ZIndex_i^q=compress(Z_i^q)$。以一阶指数哥伦布编码算法为例，本文“零”索引构建算法具体过程如算法 1 所示。

**算法 1** “零”索引构建算法

输入：查询图集合 Q，压缩阈值 l

输出：“零”索引序列 ZIndex

1. ZIndex=∅;
2. for i from 1 to L do $I(p_i)=\emptyset, N(p_i)=\emptyset$
3. for q∈Q do
4. for i from 1 to L do
5. $P_i(q)\leftarrow$partition q based on extension probability// 分区
6. $I(p_i), N(p_i)\leftarrow$construct index for $P_i(q)$
7. $Z_i^q=(I(p_i), N(p_i))$
8. compute $s_{gap}$
9. if $s_{gap}>l$ then// 压缩
10. $ZIndex_i^q=\sum_{i=0}^{\lfloor\frac{s_{gap}}{d}\rfloor-1}compress(Z_i^q[0:d])$
11. end if
12. if $s_{gap}\leqslant l$ then
13. $ZIndex_i^q=\sum_{i=0}^{\lfloor\frac{s_{gap}}{d}\rfloor-1}compress(\sum_{j=0}^{|Z_i^q|-2}(Z_i^q(j+1)-Z_i^q(j)))$
14. end if
15. $ZIndex=\sum_{q\in Q}\sum_{i=1}^{L}ZIndex_i^q$
16. return ZIndex

本文提出的 Z-Index 算法包括过滤和验证两个阶段。在过滤阶段，图数据集 $G$ 首先经过基于扩展概率的查询图分区和层级过滤机制，得到查询图 $q$ 的候选集合 $C_q$；然后对该索引序列进行编码压缩，完成“零”索引的构建。在验证阶段，依次计算候选集 $C_q$ 中的图 $g$ 与 $q$ 的图编辑距离，最终的结果集 $R$ 保存所有满足阈值约束的图 $g$。

例 5 假设查询图 $q$ 及其分区结果如图 4 所示，数据图如图 1 中 $g_1$ 和 $g_2$ 所示，编辑距离阈值 $\tau=2$。在一级过滤阶段，计算 $LB_1(q,g_1)=(6-5)+(7-7)=1<\tau$，$LB(q,g_2)=(6-4)+(7-4)=5>\tau$，因此 $g_2$ 被过滤。在二级过滤阶段，计算 $q$ 与 $g_1$ 的不匹配分区数为 $1<\tau$，则将 $g_1$ 加入候选集 $C=\{g_1\}$。在验证阶段计算 $GED(q,g_1)=3>\tau$，则最终结果集 $R$ 为空，即查询图 $q$ 与 $g_1$ 和 $g_2$ 的编辑距离都不在阈值约束范围内。

Z-Index 算法详细过程如算法 2 所示。

**算法 2** Z-Index 算法

输入：图数据集 G，编辑距离阈值 $\tau$，查询图集合 Q

输出：$R=\{g|GED(g,q)\leqslant\tau, g\in G, q\in Q\}$

1. //初始化
2. preC=∅, C=∅, R=∅
3. // 数据预处理
4. for q∈Q do
5. compute $|V_q|, |E_q|$
6. for g∈G do
7. $N(g)\leftarrow|V_g|+|E_g|$
8. for q∈Q do
9. for g∈G do
10. $LB_1(g,q)=\|E_g|-|E_q\|+\|V_g|-|V_q\|$
11. if $LB_1<\tau$ then
12. preC=preC∪{g} //更新预候选集
13. for q∈Q do
14. $P_i(q), N_i(p)\leftarrow$partition q based on extension probability
15. for g∈preC do
16. for p∈$P_i(q)$ do
17. if $N(g)\geqslant N(p)$ then
18. C=C∪{g}
19. end if
20. ZIndex←construct index
21. for q∈Q do
22. for g∈G do
23. compute GED(g,q)
24. if $GED(g,q)<\tau$ then
25. R=R∪{g}
26. end if
27. return R

### 3.4 算法复杂度分析

首先在程序初始前，通过遍历预先计算数据库中所有图的顶点和边标签频率，该过程的时间复杂度为 $O(|G|)$，并且在过滤过程中，可以利用得到的顶点和边标签频率实现过滤，时间复杂度为 $O(|G|\times|Q|)$。然后在图分区阶段，将查询图分区并记录分区大小，然后计算分区 $p$ 的 $s(p)$，其时间复杂度为 $O(|V_q|+|E_q|)$。最后建立 $L$ 层“零”索引并压缩，进一步计算图编辑距离的精确值，因此本文算法的时间复杂度为 $O(|Q|\times L\times(O(|V_q|+|E_q|)+O(|G|\times|Q|)))$。

由于要对查询图分区并建立索引，因此本文算法的空间复杂度为 $O(L\times|P|\times|Q|)$，其中 $|P|$ 表示分区的数量。

## 4 实验结果与分析

### 4.1 数据集

本文在 3 个数据集上进行实验，验证 Z-Index 算法的性能，并从每个数据集中随机选取 100 个数据图组成查询图集合 $Q$。统计信息如表 1 所列，各数据集的详细介绍如下。

1) AIDS[1)]：来自 NCI/HIN 发展治疗项目的病毒筛选数据集，用于发现艾滋病病毒。该数据集由 42 687 个化合物组成。

2) IMDB-MULTI[2)]：一个具有实时可视化分析功能的交互式数据和网络数据存储库。本文选取其中 1 500 个数据

1) http://dtp.nci.nih.govdocsaidsaids_data.html

2) https://networkrepository.com/IMDB-MULTI.php

图用于实验。

3）GRAPHGEN[1)]：一个合成图生成器，可用于创建大量含有标签的数据图。本文使用该合成图生成器生成 10000 个数据图。

表 1　数据集的统计信息

Table 1　Datasets statistics

| 数据集名称 | 数据集规模 | 平均$\|V\|$ | 平均$\|E\|$ |
|---|---|---|---|
| AIDS | 42687 | 25.6 | 27.5 |
| IMDB-MULTI | 1500 | 13.0 | 65.9 |
| GRAPHGEN | 10000 | 25.5 | 30.0 |

### 4.2　实验环境

本文实验的运行环境为 Intel（R）Core（TM）i7-10700 CPU @ 2.90GHz，内存为 16GB，使用 Microsoft Windows 10 64 位操作系统。开发环境为 Visual Studio 2019，开发语言为 C++。

### 4.3　评估指标

本文实验中，编辑距离阈值使用范围设置为 $\tau=\{1,2,3,4,5,6\}$，并从以下 4 个方面进行实验评估。

1）过滤能力分析：本文使用平均候选集大小$|C|$、准确率 $acc$ 和召回率 $recall$ 来评估层级过滤机制的有效性和准确性，定义如下：

$$|C|=\frac{\sum_{q\in Q}|C_q|}{|Q|} \tag{7}$$

$$acc=\frac{TP+TN}{|G|} \tag{8}$$

$$recall=\frac{TP}{TP+FN} \tag{9}$$

其中，$|Q|$ 表示查询集大小，$|C_q|$ 表示查询图 $q$ 的候选集大小，$TP=|C\cap R|$，$TP+TN$ 表示被判断正确的数据图的个数，$FN$ 表示被错误过滤的数据图个数。本质上，通过过滤条件的数据图越少，即$|C|$越小，则过滤性能越好。

2）“零”索引构建代价：包含索引构建时间与索引大小。

3）查询响应时间 $T$：表示系统对查询请求做出响应的时间，是衡量图相似性搜索算法的重要指标之一。其定义如下：

$$T=T_{\text{pindex}}+T_{\text{filter}}+|C|*T_{\text{ged}} \tag{10}$$

$$T_{\text{pindex}}=|\bigcup_{i=1}^{L}Z_i|*(T_{\text{compress}}+T_P) \tag{11}$$

其中，$T_{\text{pindex}}$是基于扩展概率的查询图分区时间以及构建“零”索引的时间开销，$T_{\text{filter}}$是层级过滤产生候选集合所用的时间，$T_{\text{ged}}$是图编辑距离的计算时间。

4）可扩展性：通过讨论算法在不同规模数据集上的候选集大小和查询响应时间的变化趋势来说明算法的可扩展性。

### 4.4　实验分析

为更好地验证 Z-Index 算法的性能，本文选取现有的主流算法 Pars[19] 和 ML-Partition[21] 作为对比算法，在上述 3 个不同规模的数据集上进行实验验证。为保证实验的公平性，避免偶然因素，本文在每个评估指标下各执行 300 次查询计算，取平均值作为最终的实验结果。

#### 4.4.1　过滤能力分析

为验证本文层级过滤机制的有效性和准确性，分别使用候选集大小$|C|$、准确性 $acc$ 和召回率 $recall$ 作为评估指标。

首先验证有效性。如图 5 所示，测试了$|C|$在不同阈值下的变化情况。其中横坐标表示阈值大小，纵坐标表示候选集中数据图的数量。根据式（7），经过 300 次查询计算显示，在所有数据集上，随着阈值的增加，候选集逐渐增大，有时甚至约等于整个数据集。由图 5 可以看出，在不同数据集上，Z-Index 算法得到的候选集最小，约为 ML-Partition 候选集的 50%，其次是 ML-Partition，候选集最大的是 Pars 算法。并且随着编辑距离阈值的增大，Z-Index 算法的候选集增长速度明显比 ML-Partition 和 Pars 慢，这表明 Z-Index 算法可以更大力度地精简候选集，减少验证阶段图编辑距离的计算次数，避免了许多无效的图编辑距离计算，验证了 Z-Index 算法层级过滤机制的有效性。

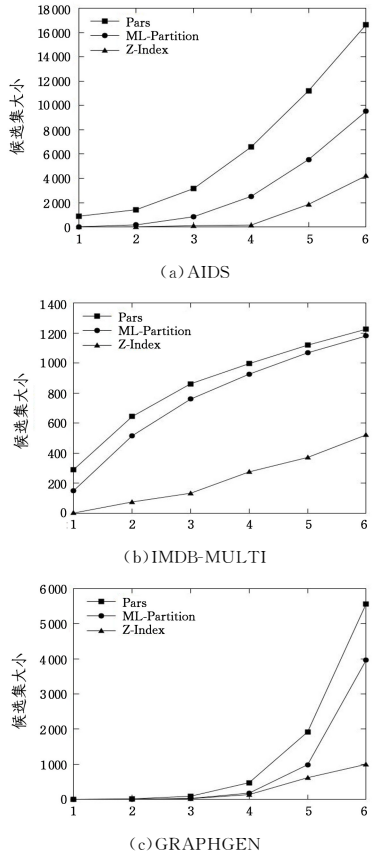



图 5　不同数据集中算法的平均候选集大小

Fig. 5　Average candidate set size of algorithms on different datasets

其次验证准确率和召回率。本部分实验固定阈值 $\tau=3$，分别统计 Pars，ML-Partition 和 Z-Index 算法产生的候选集大小$|C|$，然后对候选集中的图精确计算编辑距离，统计不在编辑距离阈值内的数据图个数，根据式（8）和式（9）计算过滤算法的准确率 $acc$ 和召回率 $recall$。如表 2 所列，Z-Index 算法得到的 $acc$ 和 $recall$ 均略高于 Pars 和 ML-Partition，其 $acc$ 最高可达 0.945，表明 Z-Index 算法可以在保证准确性的前提中得到较优的过滤效果。

1) https://www.cse.ust.hk/graphgen

表 2 不同数据集中 3 种算法的准确率和召回率

Table 2 *acc* and *recall* of three algorithms on different datasets

(单位:%)

| 数据集 | | Pars | ML-Partition | Z-Index |
|---|---|---|---|---|
| AIDS | *acc* | 0.880 | 0.917 | 0.942 |
| | *recall* | 0.891 | 0.923 | 0.950 |
| IMDB-MULTI | *acc* | 0.900 | 0.928 | 0.945 |
| | *recall* | 0.901 | 0.928 | 0.955 |
| GRAPHGEN | *Acc* | 0.890 | 0.911 | 0.939 |
| | *Recall* | 0.859 | 0.910 | 0.940 |

### 4.4.2 “零”索引构建代价分析

Z-Index 算法主要从两个方面分析索引构建代价:索引空间占用和索引构建时间。本文在编辑距离阈值 $\tau=3$ 的情况下,测试了不同编码算法下“零”索引的空间占用情况,实验结果如图 6 所示。图 6 中,$N$ 代表编码之前的索引大小,$U$ 和 $E$ 分别代表采用一元编码算法[24]和一阶指数哥伦布编码算法[24]时的索引大小。实验结果表明,在不同数据集上,采用一阶指数哥伦布编码算法具有较好的压缩效果,索引被压缩至原来的 3%左右,能更好地体现“零”索引的性能。因此,本文实验将选取一阶指数哥伦布编码算法作为 Z-Index 的编码算法。

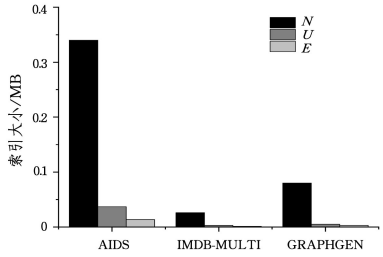



图 6 不同压缩算法下的索引占用空间情况对比

Fig. 6 Comparison of index space occupation of different compress algorithms

如表 3 所列,实验测试了在阈值 $\tau=3$ 的情况下,3 种算法在不同数据集上的平均索引大小 space 和索引构建时间 time。由表可得,在索引大小和索引构建时间上,Z-Index 算法均明显优于 ML-Partition 和 Pars。其中 Z-Index 的索引大小比 ML-Partition 和 Pars 算法中构建的索引要小得多,所用索引空间约降低了一个数量级。原因在于:一方面,查询图的数量远少于图数据集中图的数量,因此对查询图分区使得索引空间占用较少;另一方面,索引压缩算法进一步降低了索引大小。在索引构建时间方面,相较于 ML-Partition 和 Pars 算法,Z-Index 算法由于无需复杂的子图同构计算,索引构建时间最短。

表 3 不同数据集中 3 种算法的索引空间大小与索引构建时间数据统计

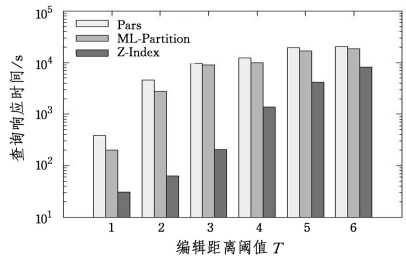
Table 3 Index space and index construction time of three algorithms

| 数据集 | | Pars | ML-Partition | Z-Index |
|---|---|---|---|---|
| AIDS | Space/MB | 17.11 | 55.42 | 0.34 |
| | Time/s | 3003.52 | 767.45 | 0.09 |
| IMDB-MULTI | space/MB | 0.72 | 2.18 | 0.03 |
| | time/s | 100.07 | 25.20 | 0.38 |
| GRAPHGEN | space/MB | 2.62 | 7.86 | 0.08 |
| | time/s | 295.31 | 195.48 | 0.65 |

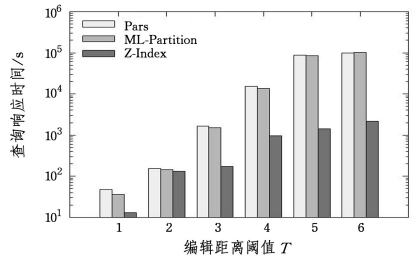
综上所述,Z-Index 算法通过对查询图分区和构建“零”索引等过程对过滤阶段做出优化,实现了算法的低索引空间占用,缩短了索引构建时间,表现出了更好的性能。

### 4.4.3 查询响应时间分析

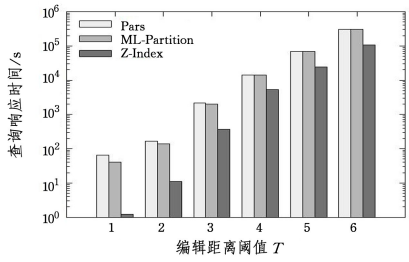
查询响应时间是衡量图相似性搜索算法性能的重要因素。在 Z-Index 算法中,查询响应时间包括基于扩展概率的查询图分区时间、“零”索引构建时间、层级过滤产生候选集的时间和图编辑距离的计算时间。由 4.4.1 节和 4.4.2 节部分的实验结果可得,Z-Index 算法在构建索引时所用时间最短,且 Z-Index 可以得到最小的候选集,使得验证阶段图编辑距离的计算时间少于 ML-Partition 和 Pars。如图 7 所示,在不同阈值 $\tau$ 下,Z-Index 算法的查询响应时间均比 ML-Partition 和 Pars 短。并且当阈值较小时,Z-Index 算法表现出了较好的效果,随着阈值的增大,其提升速度逐渐趋于平稳,查询效率的提升区间为 9.1%~78.8%。而且,不管是在稀疏图数据集 AIDS,IMDB-MULTI,还是稠密图数据集 GRAPHGEN 上,Z-Index 算法均能取得最短的查询响应时间,验证了本文算法适用于各种图数据集。




(a) AIDS




(b) IMDB-MULTI




(c) GRAPHGEN

图 7 不同数据集上算法的查询响应时间

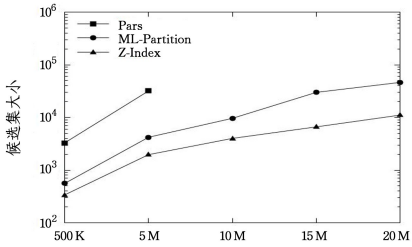
Fig. 7 Query response time of algorithms on different datasets

### 4.4.4 可扩展性分析

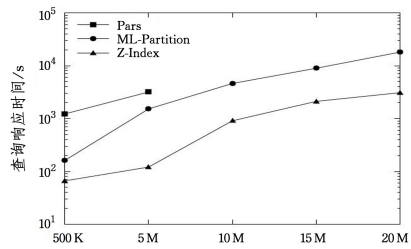
为测试并比较 Z-Index,Pars 和 ML-Partition 算法的可扩展性,本部分实验设置编辑距离阈值 $\tau$。图 8(a)记录了 3 种算法在 500K~20M 的随机数据集上的候选集大小,其中横坐标表示数据集规模,纵坐标表示候选集中数据图的个数。由图 8(a)可知,随着数据集规模的增大,Pars 和 ML-Partition

的候选集大小增长显著，这会导致验证阶段大量的图编辑距离计算。尤其对于 Pars 算法，当数据集规模超过 5M 时，程序将出现内存错误，无法得到正确结果，因此图 8 中仅给出了 Pars 在 500K 和 5M 规模上的实验数据。而 Z-Index 产生的候选集大小随数据集规模的增大增长缓慢，尤其对于 15M～20M 的大规模数据集，Z-Index 的候选集的增长速度明显比 ML-Partition 慢，表明 Z-Index 算法可扩展至较大规模的数据集，且仍具有较好的过滤效果。

图 8(b)比较了 Z-Index，Pars 和 ML-Partition 在 500K～20M 随机数据集上的查询响应时间，其中 Pars 仅在不超过 5M 的数据集上有效。由图 8(b)可知，随着数据集规模的增大，Z-Index 算法的查询响应时间始终比 Pars 和 ML-Partition 短，增长趋势也更平缓，表明数据规模的增大对 Z-Index 算法影响较小，Z-Index 即使在大规模数据集上仍具有较高的查询效率。以上实验验证了 Z-Index 算法具有良好的可扩展性。




(a)不同数据集规模上的候选集大小对比图




(b)不同数据集规模上的算法查询响应时间对比图

图 8　算法可扩展性对比图

Fig. 8　Comparison of algorithm scalability

**结束语**　本文针对图相似性搜索算法中候选集过大、索引空间占用较大和查询响应时间过长等问题，从 3 个不同的方面做出优化，提出了基于查询图分区的多层级过滤、低索引空间占用的图相似性搜索算法 Z-Index。

针对候选集过大的问题，Z-Index 算法首先基于扩展概率对查询图分区以减少图划分时间，提高过滤精度。然后采用层级过滤机制精简候选集，防止数据适应性导致的过滤误差。针对索引空间占用较大的问题，Z-Index 算法在构建索引的过程中，基于编码算法压缩索引数据结构，构建“零”索引以减少索引的空间代价，从而进一步缩短查询响应时间。在不同数据集上与 ML-Partition 和 Pars 算法的对比实验表明，Z-Index 算法能够有效解决上述问题，以较小的索引空间开销实现高效的查询性能。

但是本文算法仍然存在不足之处，未来将进一步研究如何将其扩展应用于动态图相似性搜索问题。

## 参 考 文 献

[1] ZHENG W G, ZOU L, LIAN X, et al. Graph similarity search with edit distance constraint in large graph databases[C]// ACM International Conference on Conference on Information & Knowledge Management. ACM, 2013: 1595-1600.

[2] ZHAO X, XIAO C, LIN X, et al. Efficient processing of graph similarity queries with edit distance constraints[J]. The VLDB Journal, 2013, 22(6): 727-752.

[3] ZHAO X, XIAO C, LIN X, et al. A partition-based approach to structure similarity search[J]. The VLDB Journal, 2013, 7(3): 169-180.

[4] GOUDA K, ARAFA M. An improved global lower bound for graph edit similarity search[J]. Pattern Recognition Letters, 2015, 58(Jun. 1): 8-14.

[5] WESKAMP N, HULLERMEIER E, KUHN D, et al. Multiple graph alignment for the structural analysis of protein active sites [J]. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2007, 4(2): 310-320.

[6] CHENG J, KE Y, FU W C, et al. Fast graph query processing with a low-cost index[J]. The VLDB Journal, 2011, 20(4): 521-539.

[7] WANG Y, FENG Z H, CHEN L, et al. Efficient similarity search for sets over graphs[J]. IEEE Transactions on Knowledge and Data Engineering, 2021, 33(2): 444-458.

[8] SHIMOMURA L C, OYAMADA R S, VIEIRA M R, et al. A survey on graph-based methods for similarity searches in metric spaces [J]. Information Systems, 2020, 95 (Jan.): 101507. 1-10507. 22.

[9] XIANG Y Z, TAN J X, HAN J S, et al. Survey of Graph Matching Algorithms [J]. Computer Science, 2018, 45(6): 27-31.

[10] XU Z B, ZHANG K, NING L H, et al. Summary of Graph Edit Distance[J]. Computer Science, 2018, 45(4): 11-18.

[11] HART P E, NILSSON N J, RAPHAEL B. A formal basis for the heuristic determination of minimum cost paths[J]. IEEE transactions on Systems Science and Cybernetics, 1968, 4(2): 100-107.

[12] ABU-AISHEH Z, RAVEAUX R, RAMEL J Y, et al. An exact graph edit distance algorithm for solving pattern recognition problems[C]// 4th International Conference on Pattern Recognition Applications and Methods. Lisbon, Portugal, SCITEPRESS, 2015: 271-278.

[13] ABU-AISHEH Z, RAVEAUX R, RAMEL J Y, et al. A Distributed Algorithm for Graph Edit Distance[C]// the Eighth International Conference on Advances in Databases, Knowledge and Data Applications. Lisbon, Portugal, 2016: 66-71.

[14] GOUDA K, HASSAAN M. CSI_GED: An efficient approach for graph edit similarity computation[C]// 32nd International Conference on Data Engineering (ICDE). Helsinki, Finland, IEEE, 2016: 265-276.

[15] CHANG L, FENG X, LIN X, et al. Speeding up GED verification for graph similarity search[C]// 36th International Conference on Data Engineering (ICDE). Dallas, TX, USA, IEEE, 2020: 793-804.

[16] CHEN X, HUO H W, HUAN J, et al. An efficient algorithm for graph edit distance computation[J]. Knowledge-Based Systems, 2019, 163: 762-775.

[17] WANG G, WANG B, YANG X, et al. Efficiently indexing large sparse graphs for similarity search [J]. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(3): 440-451.

[18] ZHENG W G, ZOU L, LIAN X, et al. Efficient graph similarity search over large graph databases[J]. IEEE Transactions on Knowledge and Data Engineering, 2014, 27(4): 964-978.

[19] ZHAO X, XIAO C, LIN X, et al. Efficient structure similarity searches: a partition-based approach[J]. The VLDB Journal, 2018, 27(1): 53-78.

[20] DIEGO R, RAMIRO T, POLO V. An exact approach for the multi-constraint graph partitioning problem[J]. EURO Journal on Computational Optimization, 2020, 8(3/4): 289-308.

[21] LIANG Y J, ZHAO P X. Similarity search in graph databases: A Multi-Layered Indexing Approach[C]// 2017 IEEE 33rd International Conference on Data Engineering(ICDE). San Diego, CA, USA, IEEE, 2017: 783-794.

[22] BLUMENTHAL D B, BORIA N, GAMPER J, et al. Comparing heuristics for graph edit distance computation[J]. The VLDB Journal, 2020, 29(1): 419-458.

[23] QIN J, XIAO C, WANG Y, et al. Generalizing the pigeonhole principle for similarity search in hamming space [J]. IEEE Transactions on Knowledge and Data Engineering(TKDE), 2019, 33(2): 489-505.

[24] PIBIR I G E, VENTURINI R. Techniques for inverted index compression[J]. ACM Computing Surveys, 2021, 52(6): 125: 1-125: 36.

[25] PIBIRI G E, VENTURINI R. On optimally partitioning variable-byte codes[J]. IEEE Transactions on Knowledge and Data Engineering(TKDE), 2020, 32(9): 1812-1823.

[26] GIULI O, ERMANN O, PIBIR I, et al. Clustered Elias-Fano indexes[J]. ACM Transactions on Information Systems, 2018, 36(1): 2. 1-2. 33.

**QIU Zhen**, born in 1997, postgraduate, is a member of China Computer Federation. Her main research interests include data retrieval and network security.

**ZHENG Zhaohui**, born in 1968, professor, Ph. D supervisor, is a member of China Computer Federation. His main research interests include data mining and network security.

（责任编辑：何杨）