DOI:10.19322/j.cnki.issn.1006-4710.2019.03.004

一种面向无向加权图的子图查询方法

朱 磊[1]，姚燕妮[1]，高 勇[2]，王一川[1]，姬文江[1]，黑新宏[1]，刘 征[1]

(1. 西安理工大学 计算机科学与工程学院，陕西 西安 710048；
2. 西安理工大学 自动化与信息工程学院，陕西 西安 710048)

摘要：随着图结构的大规模应用，图数据库上的查询已经成为图挖掘的研究热点。针对无向加权图，本文提出一种基于最短权值路径和拉普拉斯图谱的子图查询方法 PSQuery。首先，PSQuery 方法选取可表示数据图的高效特征属性；接着，对提取的特征属性按照哈希映射进行编码，将特征编码组合生成节点编码和图编码，并且基于图编码构建索引树；最后，通过实现过滤-验证框架得到结果集：按照提取特征的嵌套性质进行过滤生成候选集，再根据 VF2 算法进行验证得到最终的超图集合。实验结果表明，提出的方法加速了无向加权图数据库上的子图查询过程，提高了查询效率。

关键词：图挖掘；子图查询；最短权值路径；拉普拉斯图谱

中图分类号：G633.67 **文献标志码**：A **文章编号**：1006-4710(2019)03-0291-09

A subgraph querying method for undirected weighted graphs

ZHU Lei[1], YAO Yanni[1], GAO Yong[2], WANG Yichuan[1], JI Wenjiang[1], HEI Xinhong[1], LIU Zheng[1]

(1. School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China;
2. School of Automation and Information Engineering, Xi'an University of Technology, Xi'an 710048, China)

Abstract: With the wide application of graphs, the querying on graph database attracts more and more attention. This paper proposes a querying method called PSQuery based on shortest weight path and Laplacian spectra for undirected weighted graphs. Specifically, we first choose some high quality features to represent a graph, which includes the information of nodes labels, edges, shortest weight path and Laplacian spectra. Then, these information features are encoded by hash mapping, with their codes combined to build an index. Finally, we implement the filtering-and-verification framework: the false positives are pruned according to the properties of extracted features, and a candidate set is generated; the result set is obtained using the VF2 algorithm to check the candidates. The experimental results show that the proposed method speeds up the subgraph querying and improves the efficiency.

Key words: graph mining; subgraph querying; shortest weight path; Laplacian spectra

图被大规模用于构造半结构或非结构化的数据和语义。例如，化学[1]、社交网络[2]、知识图谱[3]、XML 文件[4]等。在这些应用中，实体和关系被建模为图模型，并构建更加高效的半结构或非结构化的图数据库，然后应用图挖据相关技术去高效智能地检索事物间的关联关系[5]。

图数据库相关领域中，子图查询过程包含子图同构的判定，而该问题已经证明是 NP 完全问题[6]，所以现存的方法均采用过滤-验证框架，而且提取高效的图特征进行过滤已成为子图查询的重要研究方向。一些现存的方法主要采用数据挖掘的理论去提取频繁子结构作为特征，并使用倒排序的索引进行过滤[7,8]。但是这类方法在图数据频繁更新时，必须重新挖掘频繁子结构和建立索引，导致代价过大[9]。

针对无向加权图，本文提出了一种子图查询方

收稿日期：2018-08-06

基金项目：国家自然科学基金资助项目(61602374，61602376)；陕西省自然科学基金资助项目(2016JQ6041，2017JQ6020)；陕西省教育厅科研计划资助项目(16JK1552，16JK1573)

作者简介：朱磊，男，讲师，博士，研究方向为知识图谱和图挖掘研究。E-mail：leizhu@xaut.edu.cn

法 PSQuery,提取的主要特征包括:节点标记、边、最短权值路径和拉普拉斯图谱信息。将这些信息特征分别按照不同方法进行编码,形成节点和图编码,基于图编码构建索引树进行过滤。在遍历索引树后生成候选图集合,再采用经典 VF2 算法进行子图同构验证,得到最终的结果集。通过在真实数据集和合成数据集上的实验结果对比,验证了本文所提方法提高了子图查询的效率。同时,由于 PSQuery 方法不以频繁子结构作为图的特征,因此提出的方法可以很好地处理图库频繁更新的情况。

1 相关工作

定义:(无向加权图)给定无向加权图 $G=\langle VS, ES\rangle$,其中 VS 和 ES 分别为节点集合和边集合。图中每个节点 v 可以建模为二元组 $\langle v_{\mathrm{id}}, v_{\mathrm{lable}}\rangle$,表示节点的 ID 和标记。每条边 e 建模为三元组 $\langle v_i, e_w, v_j\rangle$,用于表示节点 v_i 和 v_j 间的无向边,其边权值为 e_w。

定义:(子图查询)给定一个图数据库 D 和一个查询图 Q,其中数据集包含 n 个数据图,$D=\{G_1, G_2, \cdots, G_n\}$。子图查询的目标是在 D 中找到 Q 的所有超图集合。

在具体查询过程中,涉及的难点是查询中包含了子图同构的判定,而该问题已经被证明是 NP 完全问题。所以,现存的方法通常采用过滤-验证框架去加速子图查询过程,目的是减少子图同构判定的次数。

在子图查询的相关工作中,Closure-Tree 方法将相同子结构聚簇为闭包,并且对闭包建立索引树[10]。在遍历索引树时进行过滤,将不符合条件的闭包删除,生成候选图;在过滤时,采用子图同构的判定方法进行验证,得到结果集。但是,该方法在过滤时采用类似子图同构的判定方法进行检测,导致该方法在过滤阶段花费较高的代价[9]。

为了减少过滤阶段的判定开销,一些方法使用数据挖掘的算法,提取频繁子结构建立倒序索引。在遍历索引时,对查询图不包含的频繁子结构进行过滤,通过子图同构算法进行验证得到结果集。例如,Graphgrep[11]采用路径建立索引。而 gIndex[7]、Swiftindex[8]、FG-gindex[12]、Treepi[13]等方法挖掘频繁子图或者子树作为索引特征。这类方法在构建索引的过程中花费的代价较大,所以在图数据频繁更新时,这类算法由于必须重新挖掘频繁子结构和建立索引,导致代价过大[9]。

为了避免上述问题,第三类方法是将图的结构信息映射到数字空间,使用表示学习的方法生成编码,并且在编码的基础上建立索引。这类方法包括了 GCoding[9]和 LsGCoding[14]方法。但是,这些方法提取的结构特征或者缺失环路信息(例如 GCoding 提取的树形结构),或者提取的边不包含权值信息(例如 LsGCoding 只能提取无权边的图谱),这些都会在处理无向加权边的图数据库时,导致查询的性能降低[14]。

针对无向加权图数据库,G-CORE 方法使用路径信息特征,提高了无向加权图的推理和计算的效率[15]。受此启发,本文将加权路径信息和拉普拉斯图谱信息提取出来,按照表示学习方法的思路,提出了无向加权图的子图查询方法 PSQuery,来提高无向加权边图集的查询效率。

2 PSQuery 编码和索引方法

本文提出的 PSQuery 方法的处理流程如图 1 所示。

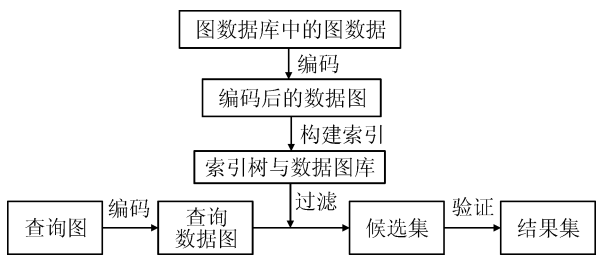


图 1 过滤-验证框架

Fig. 1 Filtering-and-verification framework

不同于 GCoding 和 LsGCoding 方法,本文将最短权值路径和图谱作为新特征去加速无向加权图的处理过程。首先,对图数据库中每个图的节点、边、最短权值路径和拉普拉斯图谱进行提取,并且将其编码生成数据图码,同时建立索引树和数据图库。接着,对查询图进行编码,生成对应的查询数据图码。然后,按照过滤-验证框架进行查询处理。需要注意的是,PSQuery 方法不仅在过滤过程中使用了新的特征编码比较,并且在验证过程中将最短路径信息也作为判定规则,目的是提高方法的整体查询效率。

2.1 图谱及路径的相关性质

定义:(邻接矩阵和 Laplacian 矩阵)给出一个无向加权图 G,其邻接权值矩阵和 Laplacian 矩阵表示为 $\boldsymbol{W}_G$ 和 $\boldsymbol{L}_G$。具体的构成为:

$$w_{(i,j)}=\begin{cases}e_w, & v_i \text{ 和 } v_j \text{ 相连时}\\ 0, & \text{其他情况}\end{cases}$$

$$l_{(i,j)}=\begin{cases}v_i \text{ 的度}, & \text{当 } i=j \text{ 时}\\ -1, & v_i \text{ 和 } v_j \text{ 相连}\\ 0, & \text{其他情况}\end{cases}$$

其中,e_w 为相连节点之间的路径权值。根据上

述矩阵的定义,可以计算出图 G 中的最短权值路径矩阵和对应的图谱信息。

定义:(图谱和 Laplacian 图谱)给出一个无向加权图 G,对应的 Laplacian 矩阵 $\boldsymbol{L}_G$ 的特征值序列称为 G 的 Laplacian 图谱。

对于无向加权图,应用权值路径信息和 Laplacian 图谱的性质,即路径权值和图谱的嵌套定理可以进行过滤操作[16,17]。

定理:(路径权值的嵌套定理)给出两个无向加权图 G_1 和 G_2,G_1 是 G_2 的子图,并且 G_1 中任意两个节点 v_i 和 v_j,G_2 中与之对应的节点为 v'_i 和 v'_j。如果 v_i 和 v_j 之间的最短权值路径为 α_k,v'_i 和 v'_j 之间的最短权值路径为 α'_k,那么它们的路径的权值应满足 $\alpha_k \geqslant \alpha'_k$。

定理:(图谱的嵌套定理)给出两个无向加权图 G_1 和 G_2,其 Laplacian 矩阵分别表示为 $\boldsymbol{LA}_{m\times m}$ 和 $\boldsymbol{LB}_{n\times n}$($m \leqslant n$)。其中 $\boldsymbol{LA}_{m\times m}$ 的特征值为 $\lambda_{m-1} \leqslant \lambda_{m-2} \leqslant \cdots \leqslant \lambda_1 \leqslant \lambda_0$,$\boldsymbol{LB}_{n\times n}$ 的特征值表示为 $\beta_{n-1} \leqslant \beta_{n-2} \leqslant \cdots \leqslant \beta_1 \leqslant \beta_0$。如果 G_1 是 G_2 的子图,那么它们的特征值满足 $\lambda_k \leqslant \beta_k$,$k=0,1,\cdots,m-1$。

基于统计学规律,无向加权图中节点标记和边的个数也具有相似的性质,即通过数值比较进行过滤操作。在 GCoding 和 LsGCoding 方法中,这两种基本特征已被提取为图的特征编码[9,14]。根据特征进行编码的操作效率高,所以 PSQuery 提取这 4 个图属性作为索引特征。

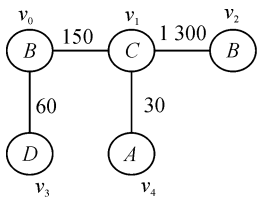
2.2 数据图的编码

本文提取的 4 个特征包含:节点标记、边、最短权值路径和拉普拉斯图谱。这 4 个特征编码的组合信息较全面地描述了无向加权图的基本信息、加权路径信息和拓扑信息。

图的编码过程是对每个原始的图数据的节点进行特征提取,生成对应编码;再将每个节点的编码组合生成图的编码。

在编码过程中,提取的第一个特征是节点的标记信息。提取到节点标记信息后,按照哈希映射函数将节点的标记映射到数字空间,生成节点标记编码;然后将所有节点标记编码合并生成图的节点标记编码。

在图 2 中,Q 包含标记为 A、C、D 的节点各一个,两个标记为 B 的节点。查找节点哈希函数,v_4 的节点标记编码为“0001”。对其他节点进行相同的操作,可以得到图中其他节点的标记编码,并且,按位相加即可得到图 Q 的节点标记编码为“1121”。



(a) 图Q

节点	哈希映射
A	0001
B	0010
C	0100
D	1000

(b) 节点的哈希映射

图 2 节点的哈希映射函数

Fig. 2 Vertex hash mapping function

提取的第二个特征为边信息。通过遍历节点相邻边的哈希映射函数,可以得到邻接边的编码。按位相加可得到图 Q 的边编码。图 3 给出图 Q 中边的哈希映射函数,其中“$*$”表示经过的任意边权值。

<*, A>	0001
<*, B>	0010
……	……
<*, D>	1000

图 3 边的哈希函数

Fig. 3 Edge hash map

对于加权图 Q 中节点 v_4 的邻接边,通过查找哈希映射函数得到边编码“0100”。最后,将 5 个节点对应的 5 个计数串按对应位相加,得到无向加权图 Q 的边编码为“1331”。

区别于现存的基于表示学习的方法,本文提取路径权值信息和拉普拉斯图谱作为特征进行映射编码。根据路径权值的嵌套定理,可以将两个节点之间的路径权值数进行比较和过滤。本文将最短权值路径作为第三个特征,进行加速过滤操作。在提取最短权值路径时,首先生成邻接权值矩阵,接着针对无向加权图,使用优化的 Floyd 算法求解各个节点之间的最短权值路径矩阵[18],如表 1 所示,其中 I_{vaule} 为初始权值。

表 1 最短权值路径矩阵的算法(算法 1)

Tab. 1 Shortest weight path matrix algorithm

输入:图的邻接权值矩阵 $\boldsymbol{W}_G$
输出:最短路径权值的矩阵 $\boldsymbol{S}_G$
1:$\boldsymbol{S}_G=\boldsymbol{W}_G$;
2:For k = 1 to n do;
3:For i = 1 to n do;
4:$I_{\text{vaule}}=\boldsymbol{S}_G[i,k]$;
5:For j = 1 to i do;
6:$\boldsymbol{S}_G[i,j]=\min\{\boldsymbol{S}_G[i,j], I_{\text{vaule}}+\boldsymbol{S}_G[k,j]\}$;
7:$\boldsymbol{S}_G[j,i]=\boldsymbol{S}_G[i,j]$;
8:Return $\boldsymbol{S}_G$。

通过算法 1,由 PSQuery 方法可以得到最短权值路径矩阵。该矩阵中的元素为节点 v_i 到节点 v_j 的最短权值路径。图 2 中 Q 的最短权值路径矩阵为:

$$\begin{bmatrix} 8 & 5 & 12 & 4 & 8 \\ 5 & 6 & 7 & 9 & 3 \\ 12 & 7 & 14 & 16 & 10 \\ 4 & 9 & 16 & 8 & 12 \\ 8 & 3 & 10 & 12 & 6 \end{bmatrix}$$

根据上述权值矩阵,PSQuery 方法通过将矩阵中的元素进行编码,并且进行计算,得到每个节点到标记为 L' 节点的最短权值路径;再取每个节点的权值中的最小值作为图 Q 的权值路径编码。图 4 给出了权值路径编码的生成过程。

路径终点	A	B	C	D
v_0	8	8	5	4
v_1	3	5	6	9
v_2	10	12	7	16
v_3	12	4	9	8
v_4	6	8	3	12

↓

路径终点	A	B	C	D
Q	3	4	3	4

图 4 图 Q 的权值路径编码

Fig. 4 Weight path coding of graph Q

PSQuery 方法提取的最后一个特征为 Laplacian 图谱。具体生成过程中,对每个节点生成 L(为一整数)层生成图,并且计算对应图谱来表示各个节点的局部拓扑信息;再将节点的图谱进行组合,得到整个无向加权图的拓扑信息,即 Laplacian 图谱。其中,L 层生成图的算法如表 2 所示。

表 2 L 层生成图的生成算法(算法 2)

Tab. 2 L-Layer graph generation algorithm

输入:图 $G=\langle VS, ES\rangle$,$v\in VS$

输出:$G_L=\langle VS', ES'\rangle$($L$ 层生成图)

1:将 v 加入节点集合 VS'中;

2:将边集合 ES'置为空集;

3:For G 中每一个节点 v'do;

4:If v' L 步可达 v then;

5:将 v'添加到 VS';

6:For G 中每一条边 e' do;

7:If G_L 包含 e'的两个节点 then;

8:将 e'添加到 ES';

9:Return $G_L=\langle VS', ES'\rangle$。

按照算法 2,PSQuery 可以生成节点 L 层生成图;接着,按照雅克比算法求解出对应的特征值,并取最大的两个特征值作为节点的图谱;最后,将所有节点的图谱组合,生成图的 Laplacian 图谱。

根据上述提取特征和对特征进行编码的过程,可以得到节点编码和图编码,具体如图 5 所示。

节点	节点标记编码	边编码	最短路径编码				拉普拉斯图谱	
v_4	0001	0100	6	8	3	12	4.00	1.00

(a) 图Q中v_4的节点编码

图	节点标记编码	边编码	最短路径编码				拉普拉斯图谱				
Q	1121	1331	3	4	3	4	2.23	2.23	1.00	1.00	1.00
							4.10	4.10	4.00	4.00	3.00

(b) 图Q的图编码

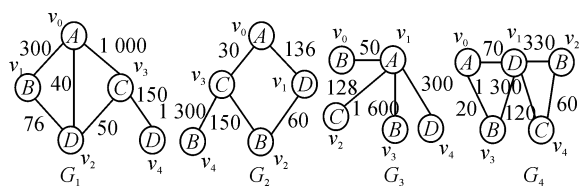
图 5 节点编码和图编码

Fig. 5 Coding of vertex and graph

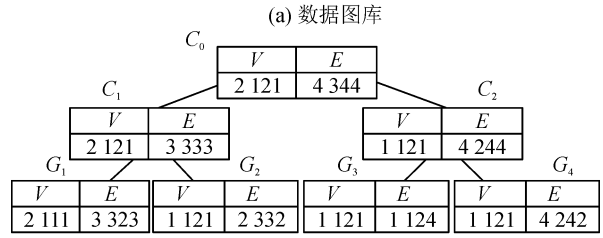
2.3 索引的建立

索引树的构建参照 GCoding 方法中索引树的构造过程[9]。使用图编码中节点标记编码和边编码作为索引树的一个节点。下面给出 4 个数据图的索引树实例。

在图 6(a)中,列出的图数据库包含了图 G_1、G_2、G_3 和 G_4。图 6(b)为该图库的索引树。PSQuery 方法提取每张数据图编码中的前两部分作为索引的特征,构建索引树。



(a) 数据图库



(b) 数据图库上的索引树

图 6 索引树

Fig. 6 Index tree

在构建索引树的过程中,对于具有相同节点标记编码 V 和边编码 E 的无向加权图,将其作为同一个叶子节点进行处理。其中,每个中间节点有 l(l 为整数)个孩子节点,且中间节点 C_0、C_1、C_2、…、C_l 的编码为 l 个孩子节点的编码值中对应位上的最大值。同时,按照平衡树的生成方法,将每个图的编码插入索引中,最终得到一个完整的平衡索引树[19]。

3 PSQuery 的查询处理

3.1 过滤过程

本文所提方法中过滤规则分为两部分：节点过滤规则和图过滤规则。具体地，先根据图过滤规则，筛选出初步符合条件的无向加权数据图集，再对这些数据图集使用节点过滤规则进行二次过滤，找出真正符合条件的候选图集。下面具体给出两个过滤规则。

图过滤规则：给定两个无向加权图 G_1 和 G_2，G_1 包含 m 个节点，G_2 包含 n 个节点。其中 $n \geqslant m$。假定它们的图编码分别为 $gCode(G_1)=\langle V(G_1), E(G_1), P(G_1), L(G_1)\rangle$ 和 $gCode(G_2)=\langle V(G_2), E(G_2), P(G_2), L(G_2)\rangle$（其中 P 为最短权值路径编码，L 为拉普拉斯图谱）。若 G_1 和 G_2 的图编码不满足下列条件：

①$V(G_1)[i]=V(G_2)[i], i=0,1,\cdots,l_V-1$；

②$E(G_1)[i] \leqslant E(G_2)[i], i=0,1,\cdots,l_E-1$；

③$P(G_1)[i] \geqslant P(G_2)[i], i=0,1,\cdots,l_V-1$；

④$L(G_1)_k[i] \leqslant L(G_2)_k[i], i=0,1, k=0,1,\cdots, m-1$。

那么 G_1 不是 G_2 的子图。

节点过滤规则：给定两个无向加权图 G_1 和 G_2，对于 G_1 中的任一节点 v，其编码为 $\langle V(v), E(v), P(v), L(v)\rangle$。如果无向加权图 G_2 中找不到这样的节点 v'，其编码对应表示为 $\langle V(v'), E(v'), P(v'), L(v')\rangle$，且满足以下条件：

①$V(v)[i]=V(v')[i], i=0,1,\cdots,l_V-1$；

②$E(v)[i] \leqslant E(v')[i], i=0,1,\cdots,l_E-1$；

③$P(v)[i] \geqslant P(v')[i], i=0,1,\cdots,l_V-1$；

④$L(v)[i] \leqslant L(v')[i], i=0,1$。

那么 G_1 不是 G_2 的子图。

由于图过滤规则的效率高于节点过滤规则，PSQuery 方法首先使用图过滤规则遍历索引树，接着使用节点过滤规则进行判定，生成规模较小的候选集。

以图 2 中的查询图 Q 为例，使用图过滤规则遍历图 6 中的索引树。首先将 Q 的图编码与根节点进行比较，发现不满足图过滤条件，继续遍历其孩子节点；当遍历到中间节点 C_2，对比编码 E 时，满足图过滤规则，则 C_2 及其孩子节点全部删除。同理，遍历到 G_1 时也满足图过滤规则，也被删除，最后产生候选集为 $\{G_2\}$。按照同样的判定方法，经过节点判定规则后，就可以生成最终的候选集合。

完成过滤后，接着对每个候选图进行验证处理，得到最终的查询结果集。

3.2 验证过程

在验证阶段，采用经典的 VF2 算法进行子图同构的判定[20]。在 VF2 方法中，对匹配对进行判定的可行性规则由两部分组成：语法规则和语义规则。语法规则按照 VF2 算法中列举的相关规则进行判定。对于语义规则，VF2 算法根据子图中节点和邻接边与超图中对应节点和对应邻接边的相似匹配关系进行判定。为了实现无向加权图中加权边的信息判定，在验证过程中，将节点过滤规则也作为语义规则的一部分进行判定，从而加速了无向加权图的同构判定效率。

4 实验结果和分析

4.1 数据源

为了进一步验证所提方法的可行性和有效性，本文选取第三类基于表示学习的 GCoding 和 LsGCoding 方法进行比较和验证。在具体的实验中，对比的两种方法均基于 iGraph 框架进行编程实现[21]。

这三种方法都提取了节点和边的信息，而实验性能差异主要是由于 GCoding 方法将生成子树的图谱作为该方法的主要特征，LsGCoding 方法将生成子图的图谱作为主要特征，而 PSQuery 方法将生成子图的拉普拉斯图谱和最短路径长度作为主要特征所导致的。由于主要特征的不同，这三个方法在过滤和验证过程中的过滤和判定效率不同，本节内容是对这三种方法的性能测试。

实验中的输入数据分为两类数据：真实数据集和合成数据集。

1）真实数据集。该数据集包含 10 000 个简单数据图作为测试数据图集，是从 Developmental Therapeutics Program 主页上已知分类的化学分子式中提取出来的，并且大部分子图查询方法均采用该数据集进行测试。其中每个图的平均节点个数为 25.4，边的个数为 27.3。输入的查询图集也来源于这个真实数据集，包括了 6 个查询集合：Q_4、Q_8、Q_{12}、Q_{16}、Q_{20}、Q_{24}，其中每个查询集 Q_i 包含了 1 000 个随机抽取的查询图，并且边的个数为 i。

2）合成数据集。合成数据集由 iGraph 提供，使用图生成器 GraphGen 生成。此数据集包含 10 000个稠密数据图，数据集中图的平均节点个数为 30，边的平均密度为 0.5。查询图集则按照真实数据集的抽取方式进行抽取，同样得到 Q_4、Q_8、Q_{12}、Q_{16}、Q_{20}、Q_{24}，一共 6 组数据集。

4.2 参数设置和实验环境

在实验中,测试 3 种方法,其中每种方法的参数设置主要涉及提取图谱时构造的节点对应的 N 层生成图或者树的层数,以及最终生成图谱时选取的特征值的个数。这两个参数选用与 GCoding 相同的两个参数值,即 $N=2$,特征值个数等于 2。这是因为在 iGraph 框架中,通过实验发现,当 N 大于 2 或特征值个数大于 2 时,候选集的大小不会发生太大变化。

实验的运行环境为 Windows XP SP3 系统,CPU 为 Intel Core CPUI7-8550U,内存大小为 3.5G,开发环境为 Visual Studio 2010。

4.3 结果评测标准

实验包含两大部分:生成编码,构建索引树的过程和查询处理过程,所以评判标准也分为两个部分。评判标准 1 为索引的构建时间和索引的大小;评判标准 2 为候选集的大小和查询时间。对应的评判结论为:构建索引和编码的时间越少,并且索引树越小,说明方法性能越好;候选集越小,查询时间越短,说明方法的查询性能越好。

4.4 实验结果与分析

实验中,从 10 000 张真实或合成数据图中随机抽取 4 次,对应图数量分别为 2 000、4 000、6 000、8 000张,加上 10 000 张数据图,形成 5 种规模的数据集。针对这 5 个不同规模的数据集,分别进行编码,构建索引和查询处理的操作,得到图 7～图 10 所示的真实数据集上的实验对比结果,以及图 11～图 14 的合成数据集上的实验对比结果。

1) 真实数据集上的结果分析。

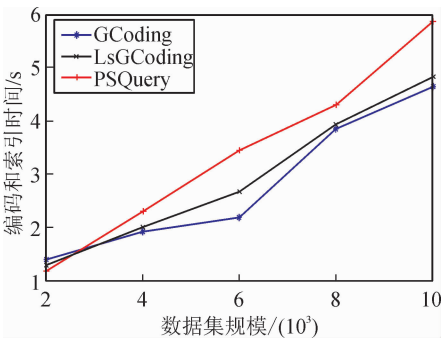


图 7 真实数据集上编码和索引时间

Fig. 7 Encoding and index-building time on real dataset

图 7 展示了真实数据图库规模从 2 000 张到 10 000张图时,3 种方法的编码和索引时间。PSQuery 方法提取节点和边信息、权值路径和 Laplacian 图谱,而 GCoding 和 LsGCoding 方法只有节点、边和图谱(或者拉普拉斯图谱)信息,所以 PSQuery 方法编码和索引构建的时间大于 GCoding 方法和 LsGCoding 方法。

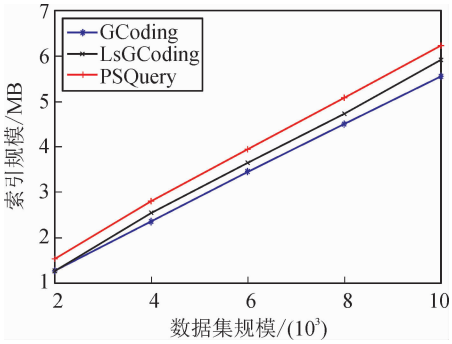


图 8 真实数据集上索引的大小

Fig. 8 Size of the index on real dataset

图 8 展示了真实数据图库从 2 000 张到 10 000 张图时,编码和索引大小的实验性能。因为 PSQuery 方法比 GCoding 和 LsGCoding 方法提取的特征多了权值路径信息,所以 PSQuery 方法中数据图库和索引所占空间更大。而 GCoding 方法的主要特征只有子树的特征值,所以 GCoding 方法的索引空间最小。

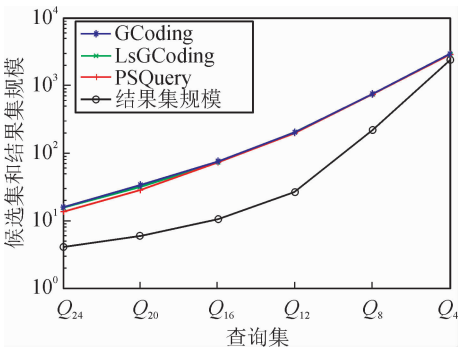


图 9 真实数据集候选图集合的大小

Fig. 9 Size of candidate dataset on real dataset

图 9 为真实数据库中不同查询图集合下,3 种方法过滤后的候选集的大小。随着查询图集合从 Q_{24} 变化到 Q_4,3 种方法的结果集在增大,候选图集也在增大。与 GCoding 和 LsGCoding 两个方法相比,由于 PSQuery 方法提取更多的权值路径信息,其裁剪过滤性能最好,所以在真实数据集上,PSQuery 方法生成的候选集最小。由于候选集规模直接影响查询效率,所以这也说明 PSQuery 方法在一定程度上提升了过滤的性能。

从图 10 中可以看出,随着真实数据集上查询图从 Q_{24} 变化到 Q_4,3 种方法的查询时间都在增大。这是因为候选集一直在增大,必须花费更多的时间去进行子图的验证操作才能得到最终结果集。并且除了 Q_4,PSQuery 方法的查询时间均小于 GCoding

方法和 LsGCoding 方法。由于提取了更好的图特征，PSQuery 方法候选集最小，验证时间最少，相应的查询时间最少，这说明 PSQuery 方法提高了真实数据集上的加权无向图的子图查询效率。

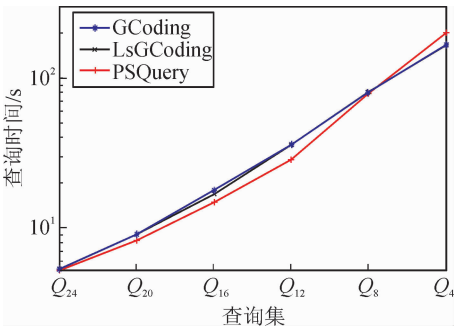


图 10 真实数据集上的查询时间

Fig. 10 Query time on real dataset

2）合成数据集上的结果分析。

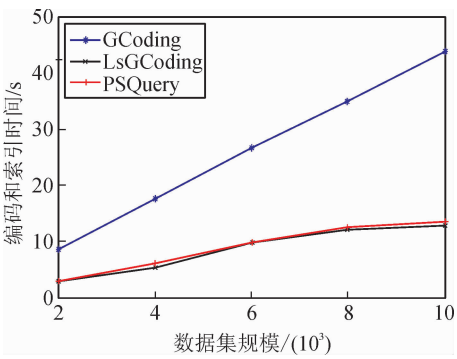


图 11 合成数据集上编码和索引时间

Fig. 11 Encoding and index-building time on synthetic dataset

图 11 给出了 3 种方法在合成数据集上的编码和索引时间。相较于 LsGCoding 和 PSQuery 方法，GCoding 方法的编码和索引时间最大。这是因为合成图大部分为稠密图，针对稠密图，GCoding 方法在生成 N 层生成树时会增加多个重复的节点，因此对应的邻接矩阵在计算特征值时会十分复杂[14]。由于 PSQuery 方法相较于 LsGCoding 方法，增加了路径的权值信息，所以花费的时间多于 LsGCoding 方法。

图 12 给出了 3 种方法在不同合成数据集规模上的索引大小。因为 PSQuery 方法提取的特征多于 GCoding 和 LsGCoding 方法，所以 PSQuery 方法的索引所占空间最大。

图 13 给出了合成数据集上 3 种方法在不同规模查询集上的候选集大小。随着查询集的边个数递减，3 种方法在合成数据集上的候选集在增大。并且稠密图中，节点和边的基本信息可以覆盖部分权值路径信息，所以在不同查询集上，3 种方法的候选集规模很接近，裁剪后的过滤性能近似相同。

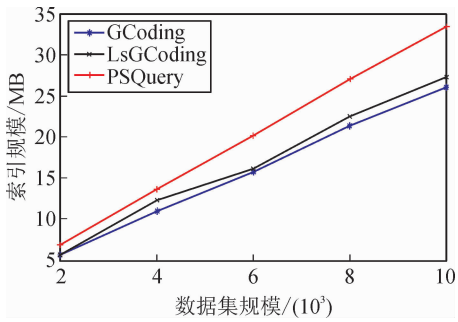


图 12 合成数据集上索引的大小

Fig. 12 Size of the index on synthetic dataset

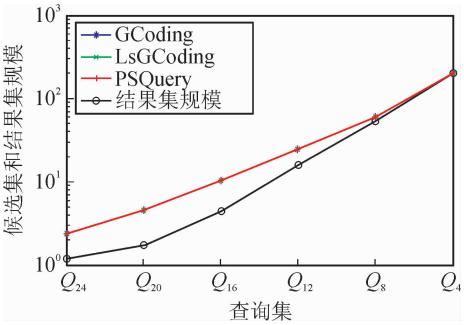


图 13 合成数据集候选图集合的大小

Fig. 13 Size of candidate dataset on synthetic dataset

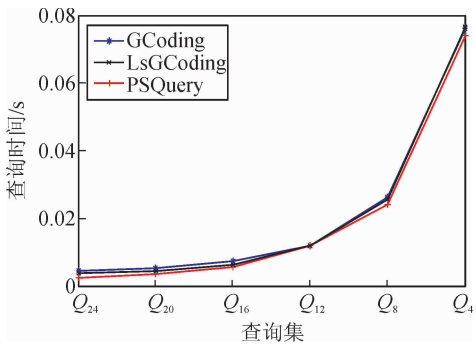


图 14 合成数据集上的查询时间

Fig. 14 Query time on synthetic dataset

图 14 给出了合成数据集上不同查询集的查询性能。当查询集从 Q_{24} 变化到 Q_4 时，LsGCoding 方法的查询时间少于 GCoding 方法。而 PSQuery 方法提取了权值路径信息，其产生的候选图集最小，所以其查询时间均小于 GCoding 和 LsGCoding 方法。

从真实数据和合成数据的实验结果可知，在编码和索引阶段，PSQuery 方法的索引时间和所占空间均大于 GCoding 和 LsGCoding 方法。但是在加权无向图的多次子图查询处理过程中，索引只需建立一次，所以索引相关操作并不能作为主要的性能评定指标。而每次查询都会产生的查询时间是其主要的性能评定指标。从真实数据集和合成数据集的查

询性能上分析,PSQuery 方法产生的候选集规模最小或者相似,并且验证过程中增加了权值的判定条件,所以 PSQuery 方法的查询时间都优于同类其他方法,从而在一定程度上提高了加权图的子图查询效率。

3) 数据集更新时的性能分析。

数据集更新的操作主要是对图数据集合进行添加操作。针对数据图集合的添加,使用 GCoding 方法提到的数据更新性能分析方法,对真实数据集,在数据集规模为 2 000 的基础上,每次增加 2 000 个数据图进行更新性能分析[9]。同样,选择经典的基于频繁子结构方法 gIndex 与 PSQuery 方法进行比较[9]。

基于表示学习的方法,在计算新插入数据图的特征编码的基础上,将其编码插入到索引树中,这样便完成了更新操作。但是,对于基于频繁子结构的方法的图更新处理,现存的处理方法有两种机制:①直接将新加入的数据图进行处理,只是在倒排索引中增加新图的 ID;②将新加入的图和原数据集重新开始进行频繁子结构挖据,新建整个倒排索引[13]。所以,在该部分实验中,将 PSQuery 方法的更新操作和 gIndex 方法的两种操作进行比较。

为了测试提出的方法在数据集更新时的性能,将 gIndex 方法和 PSQuery 方法的过滤效率和索引构建时间进行比较。采用 GCoding 方法中裁剪效率的定义:$\frac{dataset\_size - candidate\_size}{dataset\_size - result\_size}$,其中,*dataset_size* 表示数据库数据图的规模,*candidate_size* 表示候选图规模,*result_size* 表示结果图规模。该参数主要反映数据库中过滤掉的数据图集规模和实际需要过滤掉的数据图集规模的比值。其比值越高,裁剪的效率越高,适应数据集更新的性能就越好。

还有一个数据集更新时的评价指标:索引构建时间,用于描述编码的索引更新的时间。索引构建时间越少,说明更新的代价越小,适应数据集更新的性能就越好。图 15~16 为过滤效率的实验结果。

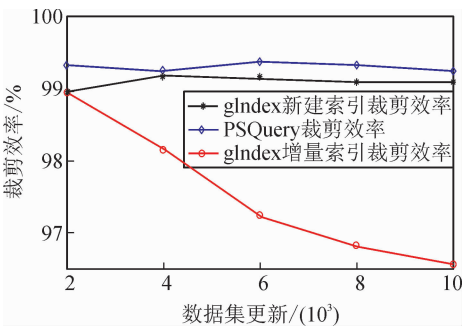


图 15 数据集更新时的裁剪效率

Fig. 15 Clipping efficiency in dataset updating

在图 15 中,PSQuery 方法在数据集每次增加 2 000个图集的情况下,裁剪的效率性能比较稳定。gIndex 方法的新建索引的裁剪效率也比较稳定,但是该方法在增量构建索引的情况下,其裁剪效率明显在下降。

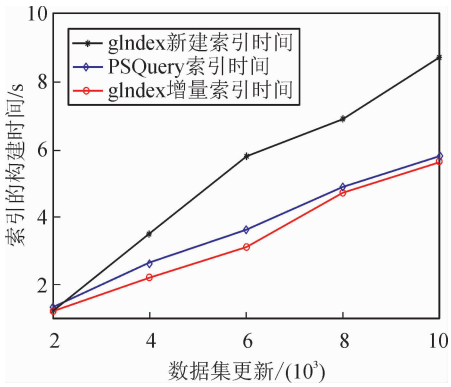


图 16 数据集更新时索引构建时间

Fig. 16 Index-building time in dataset updating

在图 16 中,随着更新数据集的增大,PSQuery 方法和 gIndex 方法构建索引的时间都在增大。对于 gIndex 方法,其增量形式构建索引方法的花费时间虽然较小,但是图 16 中该机制下索引的裁剪效率在降低,进而降低该方法的查询效率。而 gIndex 如果每次新建索引,其索引构建时间会大幅增大,而其裁剪效率变化不是很大,略低于 PSQuery 方法。PSQuery 方法在数据集更新的情况下,索引构建时间和裁剪效率相较于 gIndex 方法的两种不同机制都比较稳定。所以 PSQuery 方法可以很好地处理数据集更新的情况。

5 结 语

本文针对无向加权图的子图查询问题,提出了基于最短权值路径和 Laplacian 图谱的新编码方法,并且在编码的基础上生成了新的索引树。同时,按照过滤-验证框架进行子图查询,通过大量的实验证实了该方法具有一定的可行性和有效性,特别是 PSQuery 方法可以很好地处理数据集更新的情况。后期计划将这些新编码应用到超图查询和知识图谱的查询问题中,并且寻找更优的图特征去处理有向图的查询问题。

参考文献:

[1] ILIC A, ILIC M. On some algorithms for computing topological indices of chemical graphs [J]. Match-Communications in Mathematical and in Computer Chemistry, 2017, 78(3): 665-674.

[2] KIM J, HASTAK M. Social network analysis: charac-

teristics of online social networks after a disaster[J]. International Journal of Information Management, 2018, 38(1): 86-96.

[3] PENG P, ZOU L, ÖZSU M T, et al. Processing SPARQL queries over distributed RDF graphs[J]. The International Journal on Very Large Data Bases, 2016, 25(2): 243-268.

[4] LI N, BAI L Y. Transforming fuzzy spatiotemporal data from relational databases to XML[J]. IEEE Access, 2018, 6: 4176-4185.

[5] 漆桂林, 高桓, 吴天星. 知识图谱研究进展[J]. 情报工程, 2017, 3(1): 4-25.
QI Guilin, GAO Huan, WU Tianxing. The research advances of knowledge graph [J]. Technology Intelligence Engineering, 2017, 3(1): 4-25.

[6] MCCREESH C, PROSSER P, TRIMBLE J. Heuristics and really hard instances for subgraph isomorphism problems[C]//25th International Joint Conference on Artificial Intelligence, July 9-15, 2016, New York, USA. 2016: 631-638.

[7] YAN X F, YU P S, HAN J W. Graph indexing: a frequent structure-based approach[C]//Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, June 13-18, 2004, Paris, France. ACM, 2004: 335-346.

[8] SHANG H C, ZHANG Y, LIN X M, et al. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism[J]. Proceedings of the VLDB Endowment, 2008, 1(1): 364-375.

[9] ZOU L, CHEN L, YU J X, et al. A novel spectral coding in a large graph database[C]//Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology, March 25-29, 2008, Nantes, France. ACM, 2008: 181-192.

[10] HE H H, SINGH A K. Closure-tree: an index structure for graph queries[C]//22nd International Conference on Data Engineering (ICDE'06), April 3-7, 2006, Atlanta, GA, USA. IEEE, 2006:38-46.

[11] GIUGNO R, SHASHA D. Graphgrep: a fast and universal method for querying graphs[C]//Object Recognition Supported by User Interaction for Service Robots. IEEE, 2002, 2: 112-115.

[12] CHENG J, KE Y P, NG W K, et al. Fg-index: towards verification-free query processing on graph databases[C]//Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, June 12-14, 2007, Beijing, China. ACM, 2007: 857-872.

[13] ZHANG S J, HU M, YANG J. Treepi: a novel graph indexing method[C]// 2007 IEEE 23rd International Conference on Data Engineering, April 15-20, 2007, Istanbul, Turkey. IEEE, 2007: 966-975.

[14] ZHU L, SONG Q B. A study of Laplacian spectra of graph for subgraph queries[C]//2011 IEEE 11th International Conference on Data Mining, December 11-14, 2011, Vancouver, BC, Canada. IEEE, 2011: 1272-1277.

[15] ANGLES R, ARENAS M, BARCELÓ P, et al. G-CORE: a core for future graph query languages[C]// Proceedings of the 2018 International Conference on Management of Data, Jun 10-15, 2018, Houston, TX, USA. ACM, 2018: 1421-1432.

[16] FULLER L E. Basic matrix theory[M]. New York: Courier Dover Publications, 2017.

[17] 解阳阳, 黄强, 李向阳,等. 基于非负矩阵分解原理的方案优选方法及其应用[J]. 西安理工大学学报, 2017, 33(2): 138-144.
XIE Yangyang, HUANG Qiang, LI Xiangyang, et al. Method for optimal selection of schemes based on the non-negative matrix factorization principle and its applications[J]. Journal of Xi'an University of Technology, 2017, 33(2): 138-144.

[18] SWATHIKA O V G, HEMAMALINI S. Prims aided Floyd Warshall algorithm for shortest path identification in microgrid[J] Emerging Trends in Electrical, Communications and Information Technologies, 2017, 394: 283-291.

[19] TOUSIDOU E, BOZANIS P, MANOLOPOULOS Y. Signature-based structures for objects with set-valued attributes[J]. Information Systems, 2002, 27(2): 93-121.

[20] LEE J, HAN W S, KASPEROVICS R, et al. An in-depth comparison of subgraph isomorphism algorithms in graph data bases[J]. Proceedings of the VLDB Endowment, 2012, 6(2): 133-144.

[21] HAN W S, LEE J, PHAM M D, et al. iGraph: a framework for comparisons of disk-based graph indexing techniques[J]. Proceedings of the VLDB Endowment, 2010, 3(1-2): 449-459.

(责任编辑 周 蓓)