基于改进哈夫曼编码的大规模动态图可达查询方法

丁琳琳,李正道,纪婉婷,宋宝燕
(辽宁大学信息学院,辽宁沈阳 110036)

摘 要: 随着社交网络分析、生物信息网络分析等新兴应用的涌现和计算机技术的飞速发展,图的规模迅速增长,并且频繁更新,使得对大规模动态图数据的处理需求愈加迫切.现有的面向大规模动态图的可达查询研究成果较少,尚存在索引压缩困难以及图结构待优化等问题.本文提出了一种支持大规模动态图的基于改进哈夫曼编码的可达查询处理方法(Huffman-based Label Reachability,HuffLR).该方法首先对预处理图进行结构上的两次压缩,得到双压缩图;其次,基于双压缩图提出一种前缀label索引,该索引能够有效表达节点间的可达关系;最后,提出双压缩图的演进和可达查询处理及优化算法,主要包括边的插入与删除、节点的插入与删除.实验表明,本文提出的基于改进哈夫曼编码的大规模动态图可达查询处理方法具有良好的可行性和有效性.

关键词: 可达查询;大规模图;动态图;哈夫曼编码;标签索引
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2017)02-0359-09
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.02.014

Reachability Query of Large Scale Dynamic Graph Based on Improved Huffman Coding

DING Lin-lin,LI Zheng-dao,JI Wan-ting,SONG Bao-yan
(*School of Information,Liaoning University,Shenyang,Liaoning* 110036,*China*)

Abstract: With the emerging of applications such as Social Network Services and biological information network analysis and the rapid development of computer technology,the scale of graph increases quickly and updates frequently,so the demand to deal with large scale dynamic graph becomes more pressing. The existing research works focusing on large scale dynamic graph are rare,which have shortcomings such as difficult index compression and needing optimizing graph structure. Therefore,in this paper,we present a reachability query processing method of dynamic large scale graph based on improved Huffman Coding,named Huffman-based Label Reachability,HuffLR. Firstly,the structure of pre-processing graph is compressed twice in order to gain the double compression graph. Secondly,the prefix-label index is constructed based on the double compression graph,which can express the reachability relations between nodes effectively. Lastly,we present the evolution of the double compression graph and reachability query processing and optimized algorithms,including insertion and deletion of edge,insertion and deletion of node. Experimental results demonstrate that the reachability query processing algorithm of dynamic large scale graph based on improved Huffman Coding has good feasibility and effectiveness.

Key words: reachability query; large scale graph; dynamic graph; Huffman coding; label index

1 引言

图数据能够有效描述现实生活中各类事物之间的复杂关系.随着社交网络分析、生物信息网络分析等新兴应用的涌现和计算机技术的飞速发展,图的规模迅速增长,并且频繁更新,使得对大规模动态图数据的处理需求愈加迫切.可达查询作为图数据管理中的重要研究问题获得了广泛的关注,一直是图数据管理领域的重点和热点研究问题.目前,关于大规模动态图可达查询处理的研究成果并不多,往往通过索引实现可达查询,但大都存在索引压缩困难以及图结构待优化等问题.

收稿日期:2015-09-06 修回日期:2015-12-22;责任编辑:蓝红杰
基金项目:国家自然科学基金(No.61472169,No.61502215,No.61572119,No.61303016,No.61472069,No.11547235);辽宁省教育厅优秀人才项目(No.LR201017);辽宁省教育厅科学研究一般项目(No.L2015193);辽宁省博士科研启动基金(No.201501127);辽宁大学青年科研基金(No.LDQN201438)

针对上述问题,本文提出了支持大规模动态图的基于改进哈夫曼编码的可达查询处理方法 HuffLR(Huffman - based Label Reachability,HuffLR).首先,对预处理的图进行结构上的两次压缩,得到双压缩图;其次,为双压缩图构建前缀 label 可达索引,该索引能够有效表达节点间的可达关系;最后,为了有效支持大规模动态图中的节点与边变化,提出了双压缩图的演进和可达查询处理及优化算法,主要包括边的插入与删除、节点的插入与删除等情况.

本文提出的 HuffLR 方法具有良好的可行性和有效性,不必给节点分配多重 label 索引,既减少了存储空间,又降低了 label 索引的初始化以及更新的复杂度.另外,该算法还对图和索引进行了高度的压缩,大大减少了存储空间、处理复杂度和查询处理时图的规模.本文的贡献点归纳如下:

(1)提出双压缩图结构,首先运用 DAG(Directed Acyclic Graph,DAG)压缩方法对图进行第一次压缩;然后运用单链压缩方法对第一次压缩的结果进行单链压缩,进而降低图的存储空间和查询处理时图的规模.

(2)提出一种基于改进哈夫曼编码的前缀 label 索引及其优化,能够有效表达节点间的可达关系,大大减少了索引的存储空间.

(3)为了有效支持大规模动态图中的节点与边的变化,提出了双压缩图的演进和可达查询处理算法,使得在图数据频繁更新的过程中,只需对部分数据进行调整即可,从而降低了查询复杂度.

(4)在模拟数据集和真实数据集上进行了大量实验,验证了 HuffLR 方法在大规模动态图上实现可达查询处理的有效性和可行性,且具有良好的查询效率和较小的索引空间代价.

2 相关工作

关于图的可达查询,可分为针对静态图和动态图两种.针对静态图的可达查询,目前已经取得了丰厚的研究成果.根据所处理的图数据的规模又可以将静态图的可达查询大致分为两类:一类是支持中小规模静态图的可达查询,如 Optimal - tree[1]、Hop labeling[2~5]、Path - tree[6]、Interval labeling[7~10]等,虽然其查询效率较高,却存在索引处理花费较大和难以有效处理大规模动态图等不足.另一类是支持大规模静态图的可达查询,如 PWAH[11],虽然支持大规模图上的可达查询处理,但其索引压缩率低,不能有效处理动态图数据,查询效率较低.

针对动态图的可达查询的研究成果并不多,大多基于索引实现可达查询.根据索引的处理类型和图数据规模大致可以分为两类[12]:支持中小规模图和支持中大规模图的非受限查询动态索引.

支持中小规模图的非受限查询动态索引大致分为仅支持插入更新的 Incremental 算法、仅支持删除更新的 Decremental 算法和全支持的 Fully Dynamic 算法三类.Fully Dynamic 算法[13~15]能够有效支持点和边的插入与删除更新,处理动态图的可达查询,但是存在索引规模大、处理的图数据规模小等不足.

支持中大规模图的非受限查询动态索引目前的相关研究工作较少.以 DAGGER[16]算法为例,其通过 Tarjan 算法[17]来处理初始有向图并分配间隔标签索引,索引空间复杂度与更新复杂度都为 $O(n)$,能够支持边、节点的删除和插入更新操作.然而,此方法需要维持多重索引,索引的分配、构建相对复杂,索引更新效率低.

本文提出的支持大规模动态图的可达查询处理算法 HuffLR 对图和索引结构都进行了高度的压缩,大大减少了存储空间、算法处理复杂度和查询处理时图的规模.

3 双压缩图及其索引

3.1 双压缩图

3.1.1 DAG 压缩

定义 1 初始图($G^i=(V^i,E^i)$) 即可达查询的原始图,其中 V^i 表示初始节点集,E^i 表示初始边集.

如图 1 所示,初始图 G^i 是由 22 个节点组成的有向图,其中节点集{A,B,C}、{D,E,F,G}、{T,S,N,O,P}分别形成了一个强连通子图.由于强连通子图内所有节点之间两两可达,为了避免处理强连通子图内节点间的可达查询,本文运用强连通分量(Strongly Connected Components,SCC)的相关性质对初始图进行 DAG 压缩.如图 2 所示,DAG 压缩就是用 SCC 节点 1、2、3 分别替代 G^i 中的节点集{A,B,C}、{D,E,F,G}、{T,S,N,O,P}.因此,当查询节点 B 到 N 是否可达(即 $B \xrightarrow{?} N$)时,只查询其分别所属 SCC 节点的可达性即可(即 $1 \xrightarrow{?} 3$).

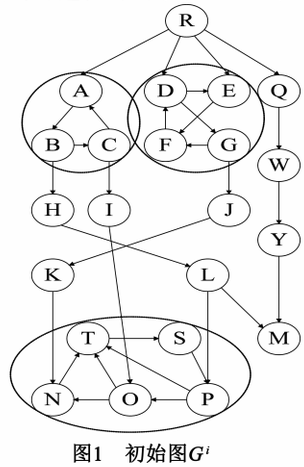

图1 初始图G^i

3.1.2 单链压缩

单链压缩是指在 DAG 压缩基础上,对于图中由 SCC 节点组成的一条单链路径(如图 1 中的节点集{Q,W,Y,M}和{{D,E,F,G},J,K}}),根据单链上 SCC 节点的"合适的"节点数 δ 来进一步压缩图. 因此,经过 DAG 压缩和单链压缩后即可得到双压缩图 – HuffLR 图. HuffLR 图是对 DAG 压缩图进一步压缩其 SCC 节点的层次图,如图 2 所示.

单链压缩只需在 DAG 压缩时或者查询 SCC 节点集合时,统计 SCC 单链上的节点数 δ. 当单链上节点数 $\delta \geqslant n$(n 为预设参数)时,根据当前最大 SCC 节点编号来分配一个新节点,即形成单链聚合节点.

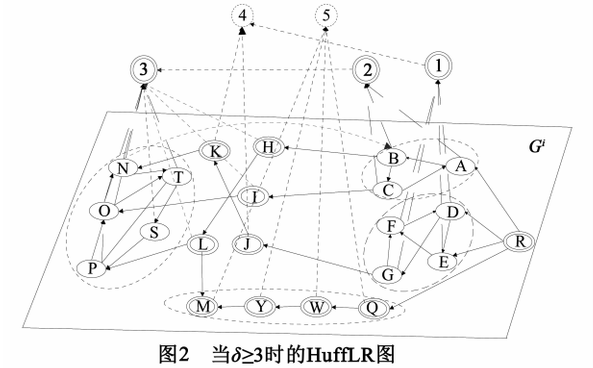

图2　当$\delta \geq 3$时的HuffLR图

如图 2,平板底部为初始图 G^i,平板上部为相应的 SCC 节点(双圈表示,节点 1、2、3)和单链聚合节点(虚线圈表示,节点 4、5). 在底部 G^i 中,对于在 V^i 中不能与其它输入节点形成 SCC 强连通子图而单个存在的节点,将其单独形成一个 SCC 节点(如平板底部双圈所示节点,节点 Q 和 I 等). 整个 HuffLR 图包含了 DAG 压缩图、G^i 和各节点间的连接边和表示节点所属关系的边(平板底部灰色虚线为添加的边(N,B)).

另外,本文使用并查集数据结构(Union-Find data structure)[18]存储平板上部与平板底部的节点间以及平板上部各节点间的所属关系,有利于存储节点和处理查询.

3.2 前缀 label 索引

本文提出的基于改进哈夫曼编码的前缀 label 索引是通过 DFS(Depth-First-Search)方法或 BFS(Breadth-First-Search)方法为每个节点赋予一个改进的哈夫曼编码的 label,其能够有效表示节点间的可达关系. 首先,给节点分配一个类似哈夫曼编码的'0'、'1'组合的 label. 然后,通过遍历其所有子节点,得出其索引为一个包含其父 label 的 label. 由于父 label 和子 label 的前面部分完全相匹配或相同,所以叫前缀匹配. 对于拥有多个父节点的节点,其 label 同时出现 0 和 1 的情况时,用 X 表示.

算法 1 表示前缀 label 索引的构建过程. 该算法仅需要在 HuffLR 图中运用 DFS 或 BFS 来分配前缀 label 索引. 第 1 行运用 DFS 算法从 root 开始访问每个节点;第 2 –6 行为遍历节点是未被访问时的处理情况. 第 3 –4 行为未访问过且为单链聚合节点时,label 索引为在父节点 L_V(当前已访问过)前缀 label 索引上分别加上'0'、$n-1$ 个'1'(其中 n 表示当前节点是第几个孩子)和($\delta-1$)个'0';第 5 –6 行为未访问过且不是单链聚合节点时,其前缀 label 索引为父节点前缀 label 索引 L_V 上分别加上'0'和 $n-1$ 个'1'. 第 7 –8 行为已访问过的节点,用新生成的 label 索引与原有 label 索引进行并操作即为新 label 索引. 第 9 –10 行将当前 label 加入标签集并返回.

算法 1　前缀 label 索引构建算法

```
Input : HuffLR
Output :L (标签集)
1   for Traversal HuffLR graph with DFS find node v do
2     if (v has not be Checked ) then
3           if ( v∈V^h)
4             L_v = L_父 + '0' + (n-1)个'1' + (δ-1)个'0';
5       else
6           L_v = L_父 + '0' + (n-1)个'1';
7     else
8         L_v = L_原 ∪ (L_父 + '0' + (n-1)个'1');
9     add L_v to L;
10  return L;
```

算法 1 中的'∪'表示从两个 label 索引首位开始对比,不同位则用综合位 X 来表示. 如 01101 和 01011,并的结果为 01XX1(X 代表任意,X = 1∪0).

图 3 表示图 2 的 label 索引分配过程,结果如图 4 所示. 首先,给 root 节点的第一个孩子节点 1 分配标签'0',然后,按 DFS 遍历节点 1 的孩子节点 H、L、3,分配'00'、'000'、'0000'. 节点 M 为 L 的第 2 个孩子,所以分配'00001',同理 I 为'001'. 当访问到 I 的孩子节点 3 时,由于已访问,得'00X0' = '0010' ∪ '0000'. 依次类推,节点 4、3、5、M 分别为'0100'、'0XX00'、'01100'和'0XX0X0'.

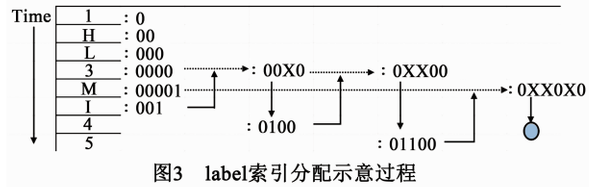

图3　label索引分配示意过程

该前缀 label 索引具有如下性质:

性质 1　假阳性:如果节点 s 可达节点 t,那么 L_s 一

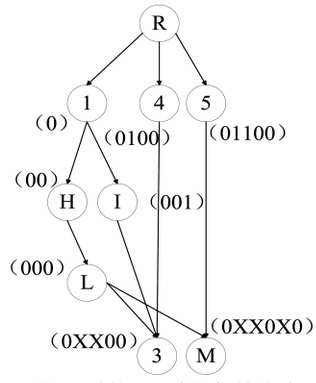

图4 当$\delta\geq3$时的label索引初始化结果图

定前缀匹配与L_t. 反之,如果L_s前缀匹配与L_t,那s不一定可达t.

运用上述前缀 label 索引即可实现可达查询,规则如下:给定两个节点u和v,首先定位其所属的 SCC 节点$s(u)$和$s(v)$;其次,判断$s(u)$和$s(v)$的 label 索引是否前缀匹配(即$s(u)$的 label 索引一定与$s(v)$的 label 索引前部分完全有效等同,并且$s(v)$中有效等同部分的下一位必须要为 0),如果不匹配,则一定不可达;否则,继续向下查询,直到找到$s(u)$的孩子到$s(v)$的路径或者提前过滤不可能的路径.

4 双压缩图的演进及可达查询

4.1 双压缩图的演进

4.1.1 边插入引起的演进

边插入即在图G^i中插入边$e=(u,v)$. 在大规模图中插入边e有可能会引起部分 SCC 节点的聚合及部分节点可达性的改变,因此需要对图进行调整. 首先对相应 SCC 结构进行调整,然后对受影响的相应节点进行 label 索引的调整.

(1)SCC 更新

由于插入边可能会导致图中部分 SCC 结构改变,需要进行图的 SCC 结构更新. 首先,定位插入边分别所属的 SCC 节点或者单链聚合节点,即$s=s(u)$,$t=s(v)$. 如果$s=t$(同属于一个 SCC 节点),无需变化;反之,则检查t是否可达s,以此证明其能否形成一个回路而形成新的 SCC 节点. 如果属于某个单链聚合节点,则分裂单链聚合节点,然后进行类似同属同一个 SCC 节点的处理.

定理 1 当两个 SCC 节点间插入一条边时,当且仅当其存在一条逆向边或逆向可达时,才可能会发生聚合.

证明:如果存在路径t到s,插入边$e=(s,t)$后就构建了一个包含路径t到s上的所有节点的回路.

如图 5 所示,当在图 1 中插入边$e=(N,B)$时,首先查找 N、B 所属 SCC 节点,得到$1=S(B)$、$3=S(N)$,且$1\neq3$,可知其等价于在 HuffLR 图中插入边$e=(3,1)$;其次,查询节点 1 到 3 是否可达,因其可能会发生聚合,把路径上访问过的节点入队 Q={1,H,L,I,3};再次,由于访问了节点 3,合并队列 Q 中节点成为一个新 SCC 节点集合,并把集合中最小编号作为当前编号,即编号 1. 节点 H、I、L 和 A、B、C 变为初始节点. 当插入边不能形成回路时,只需更新初始图G^i、HuffLR 图和前缀 label 索引即可.

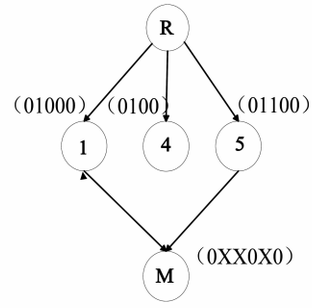

图5 对图1插入边(N,B)操作后结果图

(2)索引更新

前缀 Label 索引的更新主要是指 SCC 节点 label 的更新,大致分为聚合与不聚合两种情况.

SCC 节点不聚合:在 HuffLR 图中插入一条 DAG 边. 如果插入边的节点 label 信息已前缀匹配(即两 SCC 节点已可达),停止更新;否则,根据原 label 索引更新新的 label 索引及其所有孩子节点. 在孩子节点更新过程中,如果已经前缀匹配,停止更新,更新过程中也消除了部分的假阳性.

SCC 节点聚合:Tarjan 算法能够返回一个用来聚合成新 SCC 节点的结果集,由于结果集中的节点已相互可达,根据结果集中所有节点的父节点的 label 得出新节点的 label. 如果新 label 索引已经前缀匹配所有子节点,停止对子节点更新,否则,逐一更新或分配子节点 label 索引. 对于 SCC 单链满足$\delta\geq n$的进行单链聚合.

算法 2 为边插入算法. 第 1 行,定位插入边所在 SCC 节点集合;第 2 -5 行,处理 SCC 节点集合. 第 6 -7 行,当s、t相等时,直接加入边并更新图即可. 图 5 表示在图 1 中插入边(N,B). 其返回编号为 1 的结果集,即{1,H,L,I,3},由于结果集 1 有两个父节点 root(不考虑)和V^h节点 4 (0100),可得结果集 1 的前缀 label 索引应为 01000,再检查节点 1 的 label 与其子节点 M 是否已前缀匹配. 由于节点 M 的原 label 为 0XX0X0,已经匹配,停止更新.

算法 2 边插入算法

Input : HuffLR ,(u,v),L(标签集)
Output :HuffLR ,L(标签集)

```
find s = S(u), t = S(v);
if s≠t then
    For Traversal start with node t do
     If reach s then
         插入边,并更新所有孩子节点的前缀 label 索引
else
     直接插入边,并更新图
```

4.1.2　边删除引起的演进

边删除即在 G^i 图中删除边 $e=(u,v)$. 当在大规模图中删除边时可能会导致部分 SCC 节点分裂成许多小规模 SCC 节点. 因此需要动态更新图和前缀 label 索引来满足可达查询的要求.

(1)SCC 更新

由于删除边可能会导致图中部分 SCC 结构改变,所以需要对图的 SCC 结构进行更新. 首先,从节点 u 开始遍历,如果 u 可达节点 v,则算法结束;如果不可达,那么就会存在其他的包含节点 u 的新 SCC 节点集合,运用 Tarjan 算法来寻找新 SCC 节点,同时会遵守以下规则:1)从节点 u 开始;2)只在原 SCC 节点集合 s 中使用;3)遍历到 v 时即结束.

当找到一个新 SCC 节点集合时,会将该新节点的所有父节点入队,再对所有节点重复使用该算法. 在此过程中会统计单链上的 SCC 节点数,对于符合参数 δ 的情况会做相应的单链聚合处理.

如图 6 所示,在图 5 中删除边 $e=(L,P)$. 由于 L、P 都属于 SCC 节点 1,因此,只在 SCC 节点 1 中使用该算法即可. 首先,从 L 开始查询,L 只有孩子节点 M,但其并不属于 1,放弃,并把 L 的父节点 H 入队;其次,H 无其它子节点,接着依次将 H、B 入队,当到 B 节点时,其可以通过路径{B,C,I,O,T,S}到达 P,算法结束. 由于路径上所有节点均属于新 SCC 节点 3,而节点 H、L 则单独成为新的 SCC 节点. 由于未访问节点 A 和 N,所以其仍属于 SCC 节点 1. 对单链节点统计数 $\delta \leqslant 2$ 不做处理.

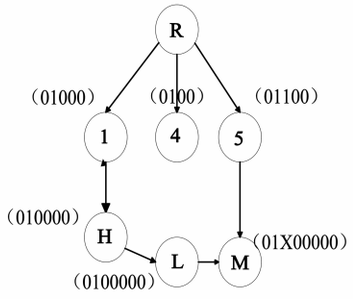

图6　在图5中删除边(L, P)后的结果

(2)索引更新

当删除边时,可根据相应新 SCC 节点或单链聚合节点的父节点来更新该节点及其所有子节点的前缀 label 索引. 由于分裂会产生新 SCC 节点,因此需要为新节点分配 label 索引.

由于在 SCC 节点集合中删除边 $e=(u,v)$ 存在两种情况,其一是保持不变,则前缀 label 索引不变;其二是 SCC 节点分裂. 针对第二种情况,由于原节点发生分裂,则需要给新增节点分配前缀 label 索引和更新原 label 索引,更新时根据该节点的父 label 来逐一更新即可,对于子 label 索引已匹配的停止更新.

算法 3 为删除边算法. 第 1 行,在并查集数据结构中定位删除边所属 SCC 节点;第 2 – 12 行,边删除时造成 SCC 节点分裂的处理;第 3 – 4 行,在原 SCC 节点中,从节点 u 开始反向访问每个父节点并将其入队;第 5 行,用 Tarjan 算法访问队里每个节点的每个孩子节点;第 6 – 8 行,如果可达 v,标记访问过的节点,形成新的 SCC 节点;第 9 – 11 行,未达目标节点 v,单独形成 SCC 节点;第 12 – 13 行,对于不属于同一 SCC 节点集合或本身为 SCC 节点时,直接删除边.

算法 3　边删除算法

```
Input :HuffLR ,G^i,(u,v),L(标签集)
Output :HuffLR ,L(标签集)
1   find s = S(u),t = S(v);
2    if s = t then
3     for Traversal each u's father node w do
4       into queue Q;
5       for Traversal each w's children p do
6          if p's next is v then
7          w out of queue Q BREAK;
8          renew label;
9        if merge a new SCC or else then
10          into queue Q;
11          assign label;
12      else
13  remove edge e = (s,t) or e = (u,v);
```

如图 6 所示,删除边 $e=(L,P)$ 时,由于 1 的父节点为 root 节点(忽略)和 4 节点,更新 1 的 label 为'01000',依次更新其孩子节点,新分裂出的 SCC 节点 H、L 分别为 010000 和 0100000,当到达 M 节点时,由于其原 label 为 0XX0X0,前缀不匹配,更新为 01X00000.

4.1.3　节点变化引起的演进

对于图中节点更新可通过边更新操作来完成,分为插入、删除节点两种操作. 当插入一个节点时,首先,插入该节点及其所有入边,根据所有父节点分配前缀 label 索引,由于该节点只有入边没有出边,所以不会形成 SCC 节点;然后,用边插入算法插入所有出边,并依次更新其所有子节点 label 即可. 当删除一个节点时,根

据边删除算法删除所有边,再删除节点即可. 但是为了减少 label 的更新次数,首先删除出边,再删除入边,即不用更新其孩子节点.

4.2 基于前缀标签索引的可达查询

HuffLR 算法是处理大规模动态图数据的可达查询方法. 该可达查询算法主要分为双压缩图的演进和可达性判断两个部分,如算法 4 所示.

算法 4 HuffLR 可达查询算法

```
Input : HuffLR,LR(修改集),L(label 集)
Output : P (节点集)
1   Batch processing the LR;
2   for each lr[i] ∈ LR
3    if lr[i] is insert edge then
4        算法 2;
5   else if lr[i] is delete edge then
6        算法 3;
7     else if lr[i] is i delete or insert node then
8     Combined with the use of 算法 2 and 算法 3;
9   if Comparison the label of nodes A and B then
10      find P;
11      return P;
```

第 1 行,根据某个时间段 $t+\Delta t$ 内接收到的点、边修改集 LR 进行批量处理,消除不必要的操作;第 2 – 8 行,对 LR 中插入和删除边、节点的情况进行双压缩图演进处理;第 9 – 11 行,根据两节点 label 判断是否前缀匹配,如已匹配,找出路径. 反之,不可达.

4.3 批量处理及索引压缩

4.3.1 批量处理

在实际应用中,查询处理操作并不是孤立存在的,常常是查询伴随着边、节点的增加与删除操作出现. 如果根据实际顺序来处理查询,会存在很多不必要的操作,如在同一个 SCC 节点集合中多次添加或删除边操作,对于同一边连续进行删除然后又复原操作等. 因此,本文对查询处理进行了批量化的操作,来进一步提高查询的效率. 在一定的查询间隔内,对接收到的处理序列进行预处理:

(1)如果存在对特定边、节点的删除和添加,将此操作从修改集 LR 中移除;

(2)如果在同一 SCC 集合内添加边,则在该 SCC 集合中处理;

(3)如果在 SCC 集合间或者 SCC 节点间删除边,则一次进行,由于假阳性的存在,可以不进行 label 索引更新操作,或多次操作用一次更新来完成;

(4)如果在 SCC 节点或 SCC 集合间插入边,可能形成更大 SCC 集合,可以一次在输入图中插入所有边,然后完成更新操作;

(5)在初始化标签阶段,为了减少重复更新节点标签,可以使用 BFS 算法一次初始化完成大多数节点的标签,以此来提高标签分配效率;

(6)在插入、删除多条边时,如果是一条完整路径,可以一次完成,减少 label 更新次数.

4.3.2 label 压缩算法

由于现实应用的图数据的规模非常大,每个节点的 label 索引也会非常大. 因此,本文提出一种对 label 索引的高度压缩算法 Extraction Compression Algorithm , ECA 算法.

由于节点 label 使用的不是单纯的二进制编码,需要将其转换成标准的二进制编码进行压缩,进而转换为十进制数存储,节省大量存储空间,如公式 1 和算法 5 所示.

$$f(l_i)=\begin{cases}A[i]=l[i],B[i]=0, & l_i=0/1\\ A[i]=0,B[i]=1, & \text{else}\end{cases} \tag{1}$$

算法 5 ECA 标签压缩算法

```
Input : L_i(label 集)
Output : L_o(label 集)
1   for each item x^i ∈ L_i
2     if x = =0 || x = =1 then
3        a[i] = x^i;
4       b[i] =0;
5     else
6        a[i] =0 ;
7        b[i] =1;
8   L_o = {a[i],b[i]};
9   return L_o;
```

算法 5 中的第 1 ~ 7 行,遍历每个节点的输入标签的每一位 X^i,如果该位为 0、1,则数组 $a[i]=X^i$,$b[i]=0$;如果该位为其它值(即为 X),那么 $a[i]=0$,$b[i]=1$;第 8 行,将该节点的两个 01 数组加入输出标签集;第 9 行,返回输出标签集 L_o. 例如,假设 $L_i=\{01XXX001010XX01100X1XX001\}$. 输出结果为 $L_o=\{a[]=\{0100000101000011000100001\},\ b[]=\{0001110000001100000101100 0\}\}$,其中,省略了数组 a 或 b 的第一位(常为 0),进而得到两个 25 位的二进制码,再用两个十进制数来表示其 label 索引即可.

相对于 DAGGER、dynamic 2-hop 算法的 label,本文只需为图中每个 SCC 节点分配、存储两个整数即可. 而 DAGGER 则需要为图中 SCC 节点分配、存储多重 label; dynamic 2-hop 则需要为每个节点存储 in 和 out 两个标签集.

5　实验测试与分析

本文通过实验来测试所提出的 HuffLR 算法的性能. 实验将本文提出的算法与 2-hop[15]、DAGGER[16] 在真实和合成数据集和不同参数下进行比较,检验算法查询效率和处理图更新时的性能.

5.1　实验环境和数据集

实验环境为四核 Intel i5-4460 3.40GHz 处理器、8GB 内存、1T 硬盘计算机,编程语言为 JAVA. 本文通过在合成数据集、真实数据集上进行测试来验证 HuffLR 算法的性能. 为了模拟现实情况,实验中通过混合更新与查询操作进行测试.

如表 1 所示设置两个数据集:其中 Phone-net 为模拟数据集,由于模拟的是某时段电话呼叫的记录网,边、SCC 节点相对较多. MicroBlog-net 为真实数据集,是部分微博转发情况的调查和数据的统计. 每个元数据由三部分构成:id, original-id, bottom-id. 节点为微博帐号, original-id 为入边(即某条微博转自哪个帐号), bottom-id 为出边(即表示本帐号中的某条微博被转到哪个帐号).

表 1　数据集信息

Dataset	Node size	Edge size
Phone-net	10,000,000	40,100,000
MicroBlog-net	22012,561	37,310,875

5.2　实验结果与分析

实验中的更新操作包括边的插入(EI)、节点的插入(NI)、边的删除(ED)和节点的删除(ND)四种处理. HuffLR 算法拥有参数 δ(单链节点聚合数),当参数 δ = 3、4、5… 时分别用 HLR3、HLR4、HLR5… 表示. 由于 DAGGER 算法在分配二重 label 索引时各项性能指标最优,因此,本实验是在 DAGGER 的二重 label 索引(DG2)时完成对比. 表 2 表示各查询算法的查询性能和更新操作性能,相对于 DAGGER 和 2-hop, HuffLR 算法的各项性能总体上表现较优.

图 9 表示在不同数据集下可达查询处理时间. 图 10 表示在不同数据集下不同算法的更新时间,其中图 10(a)表示不同算法在 Phone-net 数据集上的更新时间,图 10(b)表示不同算法在 MicroBlog-net 数据集上的更新时间. 如图 9 所示,由于 DAGGER 需要进行多重 label 的比较,2-hop 需要对标签集做交运算,而 HuffLR 算法由于SCC节点的聚合,所需进行比较的节点规模更少,所以 HuffLR 平均查询效率较好;对 HuffLR 算法来说,由于数据集 Phone-net 拥有较多边和较多 SCC 节点,其节点规模会由于双压缩而大幅度地降低,所以其在 Phone-net 数据集上的查询效率较好.

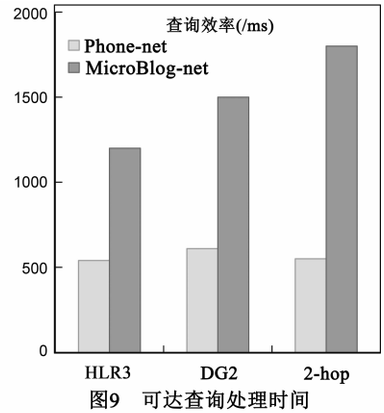

图9　可达查询处理时间

图 10 所示为更新操作的性能. 对于 EI 操作,由于此操作易导致 SCC 的聚合,且多 SCC 聚合节点也会形成 SCC 单链节点的聚合,所以对于多边时 HuffLR 和 DAGGER 的 EI 效率相当,而 2-hop 较优;对于不同数据集,由于前者节点与边多于后者,所以插入的边大部分不会影响原 SCC 节点. 但由于在 MicroBlog-net 数据集上也极易导致 SCC 节点聚合和 label 索引的更新,所以效率比 Phone-net 数据集上较差,但 HuffLR、DAGGER 总体又比 2-hop 较优.

对于 ED 操作,易导致 SCC 分裂,多边情况下对 SCC 影响较大,而 DAGGER 又要重新分配多重 label 索引,所以 HuffLR 方法占优;但 2-hop 不需要对 SCC 进行管理,HuffLR 效率又高于 2-hop;对于 HuffLR 算法,当边数较多时易造成 SCC 节点分裂而需更新和分配新的 label,而多单链情况下单链多且由于向前压缩作用而导致节点减少,所需更新较快. HuffLR 算法在多单链情况下删除边时表现出较好的效率.

对于 NI 操作,与 EI 类似,DAGGER 由于节点插入不易导致计算 SCC 的聚合,总体上 HuffLR 和 DAGGER 算法要好于 2-hop;对于 ND 操作,由于其存在假阳性和多 SCC 节点的情况,所以 HuffLR 和 DAGGER 算法较好,由于 HuffLR 算法仅更新一重 label, HuffLR 又比 DAGGER 效率较好.

表 2　各查询处理算法性能的比较(平均时间/ms)

Data	Phone-net					MicroBlog-net				
	Q	EI	ED	NI	ND	Q	EI	ED	NI	ND
HLR3	$5.4*10^2$	$1.8*10^3$	$2.2*10^3$	$2.2*10^3$	$1.2*10^3$	$1.2*10^3$	$2.8*10^3$	$3.1*10^3$	$3.8*10^3$	$1.9*10^3$
DG2	$6.1*10^2$	$2.1*10^3$	$2.8*10^3$	$2.5*10^3$	$1.4*10^3$	$1.5*10^3$	$3.6*10^3$	$3.3*10^3$	$3.9*10^3$	$2.3*10^3$
2-hop	$5.5*10^2$	$1.5*10^3$	$5.1*10^3$	$2.6*10^3$	$2.7*10^3$	$1.8*10^3$	$4.4*10^3$	$3.8*10^3$	$4.2*10^3$	$5.1*10^3$

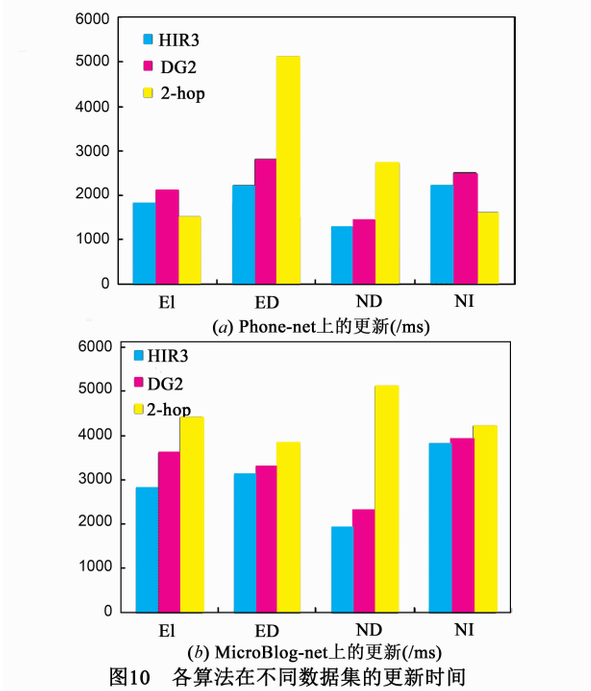

图10 各算法在不同数据集的更新时间

图 11 所示为在更新比为 $\delta=2$ 下在不同数据集上查询时间随单链聚合节点参数 δ 变化的情况. 整体趋势中间较低,两头趋于平缓. 原因是当 $\delta<3$ 时,图中会出现较多需要向前压缩处理的单链节点,过多的压缩使得压缩过程开销过大,影响查询处理效率. 同理,当 $\delta>4$ 时,虽解决了较小单链问题,但是能处理的单链节点数变少,使整个算法趋于一个较低的平稳查询效率上. 所以,当 δ 值太小时,算法处理过多不必要的节点;当 δ 值太大时,处理的节点太少,体现不出算法的价值所在,因此当 δ 在 3-5 时,算法表现出了较好的效率.

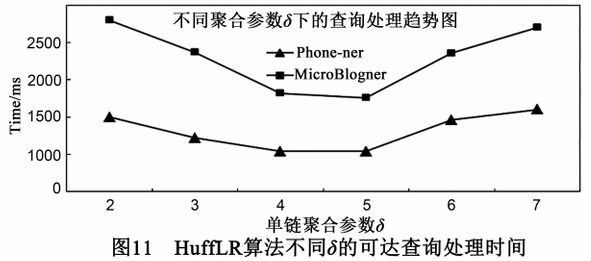

图11 HuffLR算法不同δ的可达查询处理时间

总之,HuffLR 算法非常适用于多 SCC 结构的图,其能表现出很好的查询效率,并能支持全更新操作. 同时,对多单链图也能表现出较好的性能.

6 结束语

本文将初始图通过两次压缩得到 HuffLR 图,再结合本文提出的前缀 label 索引即可实现可达查询. 不同于现有的 Interval labeling 和传递闭包方法,通过改进哈夫曼编码构造能唯一标识一条路径的特殊编码. 与现有的 DAGGER 算法相比,本文不必给每个节点分配多重 label 索引,既减少了存储空间,又降低了 label 索引的初始化、更新的次数,同时在一定程度上降低了假阳性的存在. 本文提出的前缀 label 索引及可达查询方法具有良好的性能.

参考文献

[1] R Agrawal, A Borgida, H Jagadish. Efficient management of transitive relationships in large data and knowledge bases [A]. Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data[C]. New York, NY, USA: ACM, 1989. 253 – 262.

[2] J Cheng, JX Yu, X Lin, H Wang, P S Yu. Fast computing reachability labelings for large graphs with high compression rate[A]. Proceedings of the 11th International Conference on Extending Database Technology (EBDT 2008) [C]. New York, NY, USA: ACM, 2008. 961 – 979.

[3] E Cohen, E Halperin, H Kaplan, U Zwick. Reachability and distance queries via 2-hop labels [J]. Siam Journal on Computing, 2003, 32(5): 1335 – 1355.

[4] H He, H Wang, J Yang, P S Yu. Compact reachability labeling for graph-structured data[A]. Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM 2005) [C]. New York, NY, USA: ACM, 2005. 594 – 601.

[5] R Schenkel, A Theobald, G Weikum. HOPI: an efficient connection index for complex XML document collections [A]. Proceedings of the 9th International Conference on Extending Database Technology (EDBT 2004) [C]. Berlin, Germany: Springer, 2004. 237 – 255.

[6] R Jin, N Ruan, Y Xiang, H Wang. Path-tree: An efficient reachability indexing scheme for large directed graphs[J]. ACM Transactions on Database System, 2011, 36(1): 187 – 215.

[7] Y Chen, Y Chen. An efficient algorithm for answering graph reachability queries [A]. IEEE 29th International Conference on Data Engineering (ICDE 2008) [C]. Washington, DC, USA: IEEE, 2008. 893 – 902.

[8] R Jin, Y Xiang, N Ruan, H Wang. Efficiently answering reachability queries on very large directed graphs[A]. Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD 2008) [C]. New York, NY, USA: ACM, 2008. 595 – 608.

[9] S Trissl, L Silke, U Leser. Fast and practical indexing and querying of very large graphs[A]. Proceedings of the 2007 ACM SIGMOD international conference on Management of data (SIGMOD 2007) [C]. New York, NY, USA: ACM

Press,2007. 845 – 856.

[10] H Wang,H He,J Yang,PS Yu,JX Yu. Dual labeling:Answering graph reachability queries in constant time[A]. IEEE 27th International Conference on Data Engineering (ICDE 2006)[C]. Washington,DC,USA:IEEE Computer Society,2006. 75 – 86.

[11] SJ Van Schaik,O De Moor. A memory efficient reachability data structure through bit vector compression[A]. Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data (SIGMOD 2011)[C]. New York,NY,USA:ACM,2011. 913 – 924.

[12] 富丽贞,孟小峰. 大规模图数据可达性索引技术:现状与展望[J]. 计算机研究与发展,2015,52(1):116 – 129.
Fu Lizhen, Meng Xiaofeng. Reachability Indexing for Large-Scale Graphs:Studies and Forecasts[J]. Journal of Computer Research and Development,2015,52(1):116 – 129. (in Chinese)

[13] C Demetrescu,GF Italiano. Dynamic shortest paths and transitive closure:Algorithmic techniques and data structures[J]. Journal of Discrete Algorithms,2006,4(3):353 – 383.

[14] L Roditty,U Zwick. A fully dynamic reachability algorithm for directed graphs with an almost linear update time[A]. Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing (STOC 2004)[C]. New York,NY,USA:ACM,2004. 184 – 191.

[15] AD Zhu,W Lin,S Wang,X Xiao. Reachability queries on large dynamic graphs:a total order approach[A]. Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD 2014)[C]. New York,NY,USA:ACM,2014. 1323 – 1334.

[16] H Yildirim,V Chaoji,MJ Zaki. DAGGER:a scalable index for reachability queries in large dynamic[J]. VLDB Journal,2013,21(4):509 – 534.

[17] R Tarjan. Depth-first search and linear graph algorithms[J]. Symposium on Switching & Automata Theory,1972,26(2):114 – 121.

[18] GC Harfst,EM Reingold. A potential-based amortized analysis of the union-find data structure[J]. ACM Sigact News,2000,31(3):86 – 95.

作者简介

丁琳琳　女,1983 年生于辽宁阜新. 辽宁大学信息学院副教授. 研究方向为大数据管理、分布式数据管理、图数据管理等.
E-mail:dinglinlin@ lnu. edu. cn

李正道　男,1989 年生于贵州遵义. 辽宁大学硕士研究生. 研究方向为图数据管理.
E-mail:li _ zhengdao@ 163. com

纪婉婷　女,1992 年生于辽宁沈阳. 辽宁大学硕士研究生. 研究方向为图数据管理.
E-mail:jwt@ escience. cn

宋宝燕(通信作者)　女,1965 年生于辽宁开原. 辽宁大学信息学院教授、博士生导师,CCF 高级会员、ACM 会员. 研究方向为数据库理论和技术、RFID 数据流处理技术、大数据管理、图数据管理等.
E-mail:bysong@ lnu. edu. cn