软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(4):797−812 [doi: 10.13328/j.cnki.jos.004567]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于图压缩的 *k* 可达查询处理*

李鸣鹏, 高 宏, 邹兆年

(哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

通讯作者: 李鸣鹏, E-mail: lmp@hit.edu.cn

摘 要: 研究了基于图压缩的 *k* 可达查询处理,提出了一种支持 *k* 可达查询的图压缩算法 *k*-RPC 及无需解压缩的查询处理算法,*k*-RPC 算法在所有基于等价类的支持 *k*-reach 查询的图压缩算法中是最优的.由于 *k*-RPC 算法是基于严格的等价关系,因此进一步又提出了线性时间的近似图压缩算法 *k*-GRPC.*k*-GRPC 算法允许从原始图中删除部分边,然后使用 *k*-RPC 获得更好的压缩比.提出了线性时间的无需解压缩的查询处理算法.真实数据上的实验结果表明,对于稀疏的原始图,两种压缩算法的压缩比分别可以达到 45%,对于稠密的原始图,两种压缩算法的压缩比分别可以达到 75%和 67%;与在原始图上直接进行查询处理相比,两种基于压缩图的查询处理算法效率更好,在稀疏图上的查询效率可以提高 2.5 倍.

关键词: *k* 可达;图压缩;等价类;查询处理;压缩比

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 李鸣鹏,高宏,邹兆年.基于图压缩的 *k* 可达查询处理.软件学报,2014,25(4):797−812. http://www.jos.org.cn/1000-9825/4567.htm

英文引用格式: Li MP, Gao H, Zou ZN. *K*-Reach query processing based on graph compression. Ruan Jian Xue Bao/Journal of Software, 2014,25(4):797−812 (in Chinese). http://www.jos.org.cn/1000-9825/4567.htm

K-Reach Query Processing Based on Graph Compression

LI Ming-Peng, GAO Hong, ZOU Zhao-Nian

(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

Corresponding author: LI Ming-Peng, E-mail: lmp@hit.edu.cn

Abstract: This paper focuses on *k*-reach query processing based on graph compression and proposes a *k*-reach query preserving graph compression algorithm *k*-RPC and a query processing algorithm which is able to query on the compressed graph without decompression. *k*-RPC algorithm is optimal among all the compression algorithms based on equivalent class which supports *k*-reach query. Considering *k*-RPC is based on a strict equivalent relation, this study further produces a linear approximate graph compression algorithm *k*-GRPC. *k*-GRPC first removes some edges from the input graph, then utilizes *k*-RPC to acquire better compression ratio. Novel linear query processing algorithms which are able to answer *k*-reach query on the compressed graph without decompression are introduced. Experiments on real datasets demonstrate that the compression ratios of these two compression algorithms can reach to 45% for sparse graphs and 75%, 67% for dense graphs. Comparing with the query processing on original graphs, the query performance on compressed graphs is better and for sparse graphs, it can be 2.5 times better.

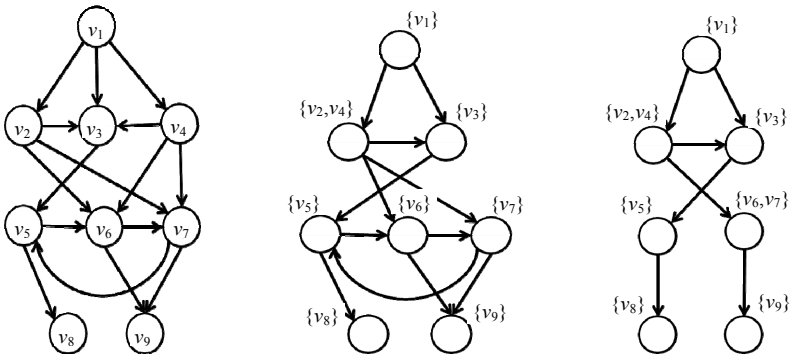
Key words: *k*-reach; graph compression; equivalent class; query processing; compression ratio

许多现实应用中,图数据的规模都在不断地扩大.Facebook 在 2012 年 4 月公布其月活跃用户人数(顶点数)达到 9 亿,好友关系(边数)达到 1 250 亿.在这样规模的图上进行查询,将是非常耗时的.

* 基金项目: 国家重点基础研究发展计划(973)(2012CB316200); 国家自然科学基金(61190115, 61033015, 61173023); 中央高校基本科研业务费专项资金(HIT.NSRIF.201180)

收稿时间: 2013-10-09; 定稿时间: 2013-12-18

在社交网络等图数据上,人们通常关心两个顶点(例如社交网络中的人)之间是否存在一条长度小于等于 k 的路径,这种查询被称为 k 可达查询.对于有向图 $G=(V,E)$,k 可达查询就是查询给定顶点 u 和 v 的最短距离是否不超过 k.例如在图 1(a)中,给定查询"顶点 v_1 和 v_8 是否 2 可达",通过计算可知,顶点 v_1 和 v_8 的最短距离为 3,所以查询结果为"否".



(a) 原始图 G (b) 压缩图 G_R (c) 压缩图 G_R^-

Fig.1 Instances of graph compression

图 1 图压缩实例

在 k 可达查询处理方面已经开展了一些研究工作:

1) 对于固定的 k 值,文献[1]中提出了一种基于建立 h 步顶点覆盖索引的 k 可达查询处理算法,可以对外存上的图数据进行 k 可达查询处理,但是这种方法只能回答特定 k 值下的 k 可达查询,对于非特定的 k 值则需要建立一组索引;
2) 图上的可达查询已经有了广泛的研究[2−4],文献[5]中也做出了较为详细的综述.因为可达查询是 k 可达查询的一种特例,即,k 为∞的情况,所以已有的可达查询处理算法不能适用于 k 可达查询;
3) 距离查询是 k 可达查询的一种扩展形式,可以通过 Dijkstra 算法实现.因为大图上的查询往往是很耗时的[6],所以文献[7–9]中都是通过建立有效的索引来实现快速的查询处理.但是,随着图规模的扩大,索引的大小也会迅速增大而无法适用.

本文用图压缩以及压缩图上直接进行查询处理的技术来解决 k 可达问题,其思想如下:首先,将图 G 进行压缩,得到一个规模很小的压缩图 G_R;然后,对于任意 G 上的 k 可达查询 Q,将 Q 转换成 G_R 上的等价查询 Q',使得 $Q(G)=Q'(G_R)$,即,在 G 上对 Q 的查询结果与在 G_R 上对 Q'的查询结果完全相同.由于 G_R 的规模远小于 G 的规模,故可以大幅提高查询处理效率,并节省存储空间.

文献[10–13]中提出了支持可达性查询的图压缩算法,其中,文献[11–13]提出了基于传递闭包的压缩算法:将原始图 $G=(V,E)$压缩为 $G_t=(V,E_t)$,满足 $E_t\in E$,且对于任意的顶点 u 和 v,u 与 v 在 G 中存在一条路径当且仅当 u 与 v 在 G_t 中存在一条路径;文献[10]则通过将具有相同祖先和后继的顶点$\{u_1,u_2,\ldots,u_l\}$压缩为一个等价类$[u]_R$,实现了对原始图的有效压缩.由于上述压缩算法仅能保留顶点之间是否可达的信息而丢失了距离信息,因此无法适用于 k 可达查询.这是因为:

1) 可达查询只是 k 可达查询的一种特殊形式,即 k 值为∞的情况;
2) 在可达查询中往往并不关心顶点之间的最短距离,因此距离信息会被丢失;而在 k 可达查询中,任意两个顶点 u 和 v 之间最短距离的缺失都可能会导致查询结果的不正确.

本文首先提出了图压缩算法 k-RPC 以及压缩图 $G_R=(V_R,E_R)$上无需解压缩的 k 可达查询处理算法,其时间复杂度分别为 $O(|E|\log|V|)$和 $O(|V_R|+|E_R|)$.本文证明了 k-RPC 算法是所有支持 k 可达查询的基于等价类的图压缩算法中最优的,即,任何一个基于等价类且支持 k 可达查询的图压缩算法得到的压缩图都不会比 k-RPC 算法得到

的压缩图更小.通过实验验证:k-RPC 算法可以将稀疏的原始图压缩掉 55%,将稠密的原始图压缩掉 25%;与在原始图上直接进行查询处理相比,基于压缩图的查询处理在稀疏图上的效率可以提高 2.5 倍.

由于 k-RPC 是基于比较严格的等价关系,当原始图比较稠密时,该等价关系将变得不易满足.此时,可以通过删除原始图 G 中的一些边以使 G 变得稀疏,然后使用 k-RPC 进行压缩.由于删除的边集 E^+将会影响到压缩图的大小,故,本文提出了最优边集问题来计算使压缩图最小的 E^*.本文进一步证明了最优边集问题是 NP-hard 的,并提出了线性的近似算法 k-GRPC 以及压缩图 $G_R^- = (V_R^-, E_R^-)$ 上无需解压缩的 k 可达查询处理算法.k-GRPC 算法的时间复杂度为 $O(|E|\log|V|+s|E_R|)$,查询处理算法的时间复杂度为 $O(|V_R^-|+|E_R^-|+|E^+|)$.通过实验验证:k-GRPC 算法可以将稀疏的原始图压缩掉 55%,将稠密的原始图压缩掉 33%;与在原始图上直接进行查询处理相比,基于压缩图的查询处理在稀疏图上的效率也可以提高 2.5 倍.

本文的主要工作及贡献如下:

(1) 提出了一种支持 k 可达查询的图压缩算法 k-RPC 以及压缩图上无需解压缩的查询处理算法,证明了其正确性,在稀疏图和稠密图上的压缩比分别可达到 45%和 75%,稀疏图上的查询效率可提高 2.5 倍;

(2) 证明了 k-RPC 在支持 k 可达查询的基于等价类的图压缩算法中是最优的;

(3) 提出了最优边集问题并证明了该问题是 NP-hard 的;

(4) 提出了线性的近似图压缩算法 k-GRPC 以及压缩图上无需解压缩的 k 可达查询处理算法,进一步将在稀疏图和稠密图上的压缩比提高到 45%和 67%,稀疏图上的查询效率可提高 2.5 倍.

1 相关工作

在大图上的 k 可达查询处理已经有了一些相关的研究[6−9,14].已有的针对距离查询的算法都是通过建立有效的索引来实现快速的查询处理,即采用了“空间换时间”的策略,其中,文献[6,8]提出的 2-hop label 通过建立 $O(|V|\cdot|E|^{1/2})$的索引,可以在 $O(|H|/n)$内完成查询处理,其中,$|H|$表示了索引的大小;TEDI[7]通过 $O(n^2)$的时间构造了 $O(l|R|^2)$的树结构索引,可以在 $O(tw^2h)$内完成查询处理,其中,tw 表示了树的宽度,h 表示了树的高度.但是随着图规模的扩大,这些方法建立索引的大小也会迅速增大而无法适用.

文献[1]中,针对 k 可达查询提出了一种基于建立 h 步顶点覆盖索引的方法,可以较为有效地建立索引并完成查询,但是这种方法建立的索引只能用来回答特定 k 值下的 k 可达查询.

支持可达查询的图压缩技术已被提出[10−13],其中,文献[11−13]提出了基于传递闭包的图压缩方法,可以在保留可达信息的同时删除原始图中大量的边;文献[10]通过将具有相同祖先和后继的顶点归为一个等价类,从而实现对原始图更加有效的压缩.但是因为可达查询并不关心顶点之间的最短距离,所以这些方法不能适用于 k 可达查询.

2 基于图压缩的查询处理算法

本文针对有向无权图进行研究,给定图 $G=(V,E)$,对于顶点 $u\in V,v\in V$,(u,v)和(v,u)是不同的两条边.尽管有向边通常也使用$\langle u,v\rangle$表示,在不引起歧义时,本文仍使用(u,v)表述.由于自环(u,u)不影响 G 上的 k 可达查询,因此可以从 G 中删除,故,本文仅考虑不带有自环的有向图.

对于任意的顶点 $u\in V$,$father(u)=\{w|(w,u)\in E\}$,$child(u)=\{w|(u,w)\in E\}$,本文使用的有向图采用邻接表的形式存储,且所有顶点的邻居都是按顺序存储的.文中使用 $d(G,u,v)$表示 G 中顶点 u 和 v 之间的最短距离.下面给出图 G 顶点集合 V 上等价关系“≡”的定义.

定义 1. 给定图 $G=(V,E)$,“≡”是 V 上的二元关系,对于顶点 $u\in V,v\in V,u\equiv v$ 当且仅当 $father(u)=father(v)$且 $child(u)=child(v)$.

易知,关系“≡”满足自反性、对称性和传递性,所以关系“≡”是 V 上的等价关系.可以根据“≡”将 V 划分为若干个等价类,$[u]_R=\{x|x\in V,u\equiv x\}$表示包含顶点 u 的等价类.

2.1 图压缩算法*k*-RPC及查询算法

本文首先给出一种图压缩方法 *k*-RPC:将图 G 中的顶点划分为若干个等价类,每个等价类都是压缩图 $G_R=(V_R,E_R)$中的一个顶点;压缩图 G_R 中两个顶点$[u]_R$和$[v]_R$之间存在一条边$([u]_R,[v]_R)$当且仅当在原始图 G 中存在两个顶点 $u_i\in[u]_R$ 和 $v_j\in[v]_R$,满足$(u_i,v_j)\in E$.算法 1 给出了 *k*-RPC 的具体执行过程.

Algorithm 1. *k*-RPC.

Input: Graph $G=(V,E)$;

Output: Compressed Graph $G_R=(V_R,E_R)$.

① **for** each node u **do**

② $[u]_R=\{u\}$;

③ **end for**

④ sort all $u\in V$ according to u's neighbors;

//对顶点排序

⑤ $CF=CC=CV=\varnothing$, $V_R=E_R=\varnothing$;

//CF,CC 和 CV 分别表示当前父亲集合、儿子集合和等价类

⑥ **for** each ordered $u\in V$ **do** //计算所有等价类

⑦ **If** $father(u)=CF\wedge child(u)=CC$ **then**

⑧ $[CV]_R=[CV]_R\cup u$, $[u]_R=\varnothing$;

⑨ **else**

⑩ $V_R=V_R\cup[CV]_R$;

⑪ $CF=father(u)$, $CC=child(u)$, $CV=u$;

⑫ **end if**

⑬ **end for**

⑭ **For** each vertex pair $[u]_R\in V_R$, $[v]_R\in V_R$ **do**

//计算压缩图的边集

⑮ **if** $(u,v)\in E$ **then**

⑯ $E_R=E_R\cup([u]_R,[v]_R)$;

⑰ **end if**

⑱ **end for**

⑲ **return** $G_R=(V_R,E_R)$;

在将 G 压缩为 G_R 后,可以在压缩图 G_R 上直接进行 k 可达查询.对于任意给定的 G 上的两个顶点 u 和 v,为了判断 u 和 v 在 G 中是否 k 可达,可以通过判断$[u]_R$和$[v]_R$在 G_R 中是否 k 可达而给出结果.我们使用了广度优先算法来查询压缩图上任意两个顶点之间的最短距离.

例如,在对图 1(a)所示的原始图 G 进行压缩时,因为顶点 v_2 和 v_4 具有相同的父亲$\{v_1\}$和儿子$\{v_3,v_6,v_7\}$,所以将合并 v_2 和 v_4 得到图 1(b)所示的压缩图 G_R.而对于给定查询"顶点 v_4 和 v_8 是否 3 可达",根据压缩图 G_R,可以找到一条路径$[v_4]_R\rightarrow[v_3]_R\rightarrow[v_5]_R\rightarrow[v_8]_R$,故将给出查询结果"是",这与在原始图 G 上的查询结果是相同的.

以下定理将证明在压缩图 G_R 上进行 k 可达查询的结果是正确的.

引理 1. 在压缩图 $G_R=(V_R,E_R)$中,$[u]_R\in V_R$,$[v]_R\in V_R$,那么$([u]_R,[v]_R)\in E_R$ 当且仅当对于任意顶点 $u_i\in[u]_R$,$v_j\in[v]_R$,均有$(u_i,v_j)\in E$.

证明:由 *k*-RPC 计算压缩图 G_R 的方法,充分性显然.下面证明必要性.

若$([u]_R,[v]_R)\in E_R$,则存在顶点 $u_0\in[u]_R$,$v_0\in[v]_R$,使得$(u_0,v_0)\in E$.

因为等价类$[u]_R$中任意两个顶点的儿子完全相同,所以对于任意顶点 $u_i\in[u]_R$,均有$(u_i,v_0)\in E$.

又因为等价类$[v]_R$中任意两个顶点的父亲完全相同,所以对于任意顶点 $u_i\in[u]_R$,$v_j\in[v]_R$,均有$(u_i,v_j)\in E$.

证毕. □

定理 1. 给定有向图 $G=(V,E)$,$G_R=(V_R,E_R)$是通过 k-RPC 算法得到的压缩图,则 G 中任意两个顶点 u 和 v 满足 $d(G_R,[u]_R,[v]_R)=d(G,u,v)$.

证明:假设 $d(G,u,v)=d$,即,在 G 中顶点 u 可以通过路径 P:$u\to w_1\to w_2\to\ldots\to w_{d-1}\to v$ 到达顶点 v.如果对于路径 P 上任意两个顶点 w_i 和 w_j 都有$[w_i]_R\neq[w_j]_R$,那么,P':$[u]_R\to[w_1]_R\to[w_2]_R\to\ldots\to[w_{d-1}]_R\to[v]_R$ 是一条连接顶点 u 和 v 的长度为 d 的简单路径;否则,P 上存在两个顶点 w_i 和 w_j,使得$[w_i]_R=[w_j]_R$.于是我们有:顶点 w_i 和 w_j 的儿子完全相同.不妨假设 $i<j$,那么顶点 u 可以通过路径 $u\to w_1\to\ldots\to w_i\to w_{j+1}\to\ldots\to w_{d-1}\to v$ 到达顶点 v.这说明 P 并不是连接顶点 u 和 v 的最短路径,矛盾.因此在 G_R 中,顶点$[u]_R$ 也一定可以通过一条长度为 d 的简单路径 P'到达顶点$[v]_R$.

下面来说明在 G_R 中,P'是连接顶点$[u]_R$ 和$[v]_R$ 的最短路径.假设 G_R 中存在一条连接顶点$[u]_R$ 和$[v]_R$ 的路径 P'': $[u]_R\to[x_1]_R\to[x_2]_R\to\ldots\to[x_p]_R\to[v]_R$ 满足 $p<d$,那么,分别从$[x_1]_R$,$[x_2]_R$,…,$[x_p]_R$ 对应的顶点集合里任意选择 1 个顶点 $y_1,y_2,\ldots,y_p$.根据引理 1 可知:对于任意 $0<i<p$,$(y_i,y_{i+1})\in E$,$(u,y_1)\in E$ 且$(y_p,v)\in E$,因此,在 G 中找到了一条连接顶点 u 和 v 的长度为 p 的简单路径 $u\to y_1\to y_2\to\ldots\to y_p\to v$.矛盾.

综上所述,我们有 $d(G_R,[u]_R,[v]_R)=d(G,u,v)$. □

下面分析 k-RPC 算法的复杂度.

因为 k-RPC 算法将判断每一对顶点 u 和 v 是否等价,根据顶点等价的定义,算法将依次访问顶点 u 和 v 的邻居.由于本文使用的有向图是采用邻接表形式存储的,且已经有序,故,判断顶点 u 和 v 是否等价的时间复杂度为 $O(deg(u)+deg(v))$.由于算法需要考察所有的顶点对,为了提高效率,算法将首先对顶点进行排序,如算法第 4 行所示.排序的规则是:根据顶点的邻居 id,以类似字典序的方法从小到大排列.例如,在图 1(a)中,顶点 v_2 的邻居是$\{v_1,v_3,v_6,v_7\}$,顶点 v_3 的邻居是$\{v_1,v_2,v_4,v_5\}$,那么,v_3 将排在 v_2 之前.容易发现,在排序后等价的顶点一定是连续排列的.因为在比较代价为 $O(1)$时,排序的复杂度为 $O(|V|\log|V|)$,而算法需要比较顶点的所有邻居,故,此时比较代价为 $O(deg(u))$,所以排序的复杂度为 $O(|V|\log|V|\cdot avg(deg(u)))=O(|E|\log|V|)$.在算法第 6 行~第 13 行,算法完成了对所有等价类的划分,此时需要访问一遍所有顶点的邻居,其代价为 $\sum_{u\in V}O(deg(u))=O(|E|)$;在算法的第 14 行~第 18 行,算法完成了压缩图边集合的构造,易知其代价为 $O(|E|)$.综上所述,k-RPC 图压缩算法的复杂度为 $O(|E|\log|V|)$.因为查询处理转化时需要计算顶点归属的等价类,我们可以在算法 1 的执行过程中记录下等价类划分的映射关系,所以查询处理转化的代价为 $O(1)$,故,使用广度优先算法进行 k 可达查询处理的代价为 $O(|V_R|+|E_R|)$.

综上所述,使用 k-RPC 算法有两个优点:

1) 可以实现对原始图 G 的有效压缩,从而节省大量的存储空间;
2) 可以在压缩图 G_R 上直接进行查询而无需进行解压缩,从而将查询处理的时间从 $O(|V|+|E|)$降低为 $O(|V_R|+|E_R|)$.

事实上,根据定理 1,原始图上任意两个顶点 u 和 v 之间的距离均与其在压缩图上对应的等价类$[u]_R$ 和$[v]_R$ 之间的距离相等.因此,可以用压缩图作为输入,使用已有的建立索引的方法[6−9]进一步提高查询速度.由于本文的主要目标在于如何通过压缩实现有效的查询,故,查询处理采用了最为直观的广度优先算法.

当原始图发生了动态更新时,可以对压缩图做出相应的更新.图上可能的更新共有 3 种:孤立顶点的插入/删除、边的插入、边的删除.下面分别讨论 3 种情况下的更新操作:1) 容易发现,孤立顶点的插入/删除不会影响压缩图中其余等价类的划分,如果压缩图中存在由孤立点组成的等价类,那么可以直接通过向该等价类中插入或从该等价类中删除顶点来实现上述更新;2) 当发生边的插入或者删除时,边两端顶点 u 和 v 的邻居都会发生改变,因此需要更新顶点 u 和 v 的排序.通过二分法,上述排序可在 $O(\log|V|\cdot avg(deg(u)))$内完成.排序后,可以在常数时间内重新划分顶点 u 和 v 所属的等价类.

k-RPC 算法同样适用于无向图,因此也可以解决许多无向图上的问题.在枚举极大团[15]、枚举三角形[16]等诸多应用中,均可使用 k-RPC 算法压缩原始图,并且能够无需解压缩地完成查询处理.

2.2 *k*-RPC最优性分析

k-RPC 算法采用了基于等价类的压缩思想,即,从原始图 G 中找到属于同一"类"的顶点并将其压缩为 G_R 中的一个顶点,因此,对于"类"的定义决定了这类方法的压缩比.下面证明了 *k*-RPC 算法在所有支持 k 可达查询的基于等价类的图压缩算法中是最优的,即,任意一种支持 k 可达查询的基于等价类的图压缩算法得到的压缩图一定不会比 G_R 更小.

我们首先形式化地给出支持 k 可达查询的基于等价类的图压缩算法 A 的描述.给定原始图 $G=(V,E)$,算法 A 需要满足:

1) 算法 A 计算出的压缩图为 $G_A=(V_A,E_A)$,其中,V_A 中的每个顶点$[u]_A$对应着原始图 G 中的一个顶点集合,边$([u]_A,[v]_A)\in E_A$ 当且仅当在 G 中存在着两个顶点 $u_i\in[u]_A$ 和 $v_j\in[v]_A$,满足$(u_i,v_j)\in E$;
2) 对于原始图 G 中给定的任意两个顶点 u 和 v,u 和 v 在 G 中是否 k 可达的查询结果与$[u]_A$和$[v]_A$在压缩图 G_A 中是否 k 可达的查询结果是相同的.

下面通过定理 2 和定理 3 分别证明:*k*-RPC 算法在所有基于等价类的图压缩算法中是顶点数最小的和边数最小的.

定理 2. 给定原始图 $G,G_R=(V_R,E_R)$是 *k*-RPC 算法得到的压缩图,A 是支持 k 可达查询的基于等价类的图压缩算法,且 A 计算出压缩图为 $G_A=(V_A,E_A)$,那么$|V_A|\geqslant|V_R|$.

证明:假设存在一种基于等价类的算法 A^*,A^*计算出的压缩图 $G_{A^*}=(V_{A^*},E_{A^*})$ 满足$|V_{A^*}|<|V_R|$,那么在 V_R 中,一定存在着两个顶点$[u]_R$和$[v]_R$,使得等价类$[u]_R$和$[v]_R$中存在着两个顶点 u_i 和 v_j 满足$[u_i]_{A^*}=[v_j]_{A^*}$;否则,V_{A^*} 中一定至少包含了$|V_R|$个顶点.

因为顶点 u_i 和 v_j 在 V_R 中对应着不同的等价类$[u]_R$和$[v]_R$,所以 u_i 和 v_j 的父亲和儿子不完全相同.假设 y 是 u_i 的儿子,且 y 不是 v_j 的儿子(反之亦可),下面考虑顶点 y 是否在$[x]_{A^*}$中:

1) 如果 $y\in[x]_{A^*}$,那么 $([x]_{A^*},[x]_{A^*})\in E_{A^*}$,否则,给定查询"顶点 u_i 和 y 是否邻接",我们将回答"否",这是错误的;但是给定查询"顶点 v_j 和 y 是否邻接",因为 $([x]_{A^*},[x]_{A^*})\in E_{A^*}$,我们将给出肯定回答,而这也是错误的;
2) 如果顶点$y\notin[x]_{A^*}$,假设$y\in[y]_{A^*}$,那么 $([x]_{A^*},[y]_{A^*})\in E_{A^*}$,否则,给定查询"顶点 u_i 和 y 是否邻接",我们将回答"否",这是错误的;那么类似地,给定查询"顶点 v_j 和 y 是否邻接",因为 $([x]_{A^*},[y]_{A^*})\in E_{A^*}$,我们将回答"是",这同样也是错误的.

上述错误说明:压缩图 G_{A^*} 不能支持 k 可达查询,矛盾.如果 u_i 和 v_j 的儿子完全相同,类似地,也可以找到 u_i 和 v_j 不同的父亲,从而推出矛盾. □

定理 3. 给定原始图 $G,G_R=(V_R,E_R)$是 *k*-RPC 算法得到的压缩图,A 是支持 k 可达查询的基于等价类的图压缩算法,且 A 计算出压缩图为 $G_A=(V_A,E_A)$,那么$|E_A|\geqslant|E_R|$.

证明:图 G 中,边通过压缩算法 A 也被划分为不同的等价类.例如,当图 1(a)中的顶点 v_2 和 v_4 被压缩为图 1(c)中的一个等价类$\{v_2,v_4\}$时,边(v_1,v_2)和(v_1,v_4)也会随之被压缩为一条边$(v_1,\{v_2,v_4\})$.由此可知,图 G 中的两条边(x_1,y_1)和(x_2,y_2)会被 A 压缩为一条边当且仅当$[x_1]_A=[x_2]_A$并且$[y_1]_A=[y_2]_A$.

假设存在一个基于等价类的图压缩算法 A^*,使得 A^*计算出的压缩图 $G_{A^*}=(V_{A^*},E_{A^*})$ 满足$|E_{A^*}|<|E_R|$,那么 G 中一定存在两条边(x_1,y_1)和(x_2,y_2),满足 $[x_1]_{A^*}=[x_2]_{A^*}$ 并且 $[y_1]_{A^*}=[y_2]_{A^*}$,但是$[x_1]_R\neq[x_2]_R$ 或者$[y_1]_R\neq[y_2]_R$;否则,$|E_{A^*}|\geqslant|E_R|$.

不妨假设$[x_1]_R\neq[x_2]_R$,因为 $[x_1]_{A^*}=[x_2]_{A^*}$,根据定理 2,可以证明压缩图 G_{A^*} 不支持 k 可达查询.因此,不存在一个基于等价类且支持 k 可达查询的图压缩算法 A^*,使得 A^*计算的 $G_{A^*}=(V_{A^*},E_{A^*})$ 满足$|E_{A^*}|<|E_R|$. □

结论 1. 给定原始图 $G,G_R=(V_R,E_R)$是 *k*-RPC 算法得到的压缩图,A 是支持 k 可达查询的基于等价类的图压缩算法,且 A 计算出压缩图为 $G_A=(V_A,E_A)$,那么$|G_A|\geqslant|G_R|$.

证明:根据定理 2 和定理 3,结论显然成立. □

3 k-RPC 算法的改进

通过结论 1,说明了在所有支持 k 可达查询的基于等价类的图压缩算法中,k-RPC 算法计算出的压缩图具有最小规模.但是我们发现,k-RPC 使用的等价关系比较严格,例如在图 1(b)中,顶点 v_6 和 v_7 具有一部分相同的父亲和儿子,但是$[v_6]_R \neq [v_7]_R$.当原始图变得稠密时,该等价关系将不容易满足.假设原始图中任意的顶点 u 都以相同的概率与其他顶点邻接,那么在一个平均度为 1 的稀疏图上,任意的顶点 u 和 v 满足 $u \equiv v$ 的概率为 $O(1/n)$;当原始图的平均度上升为 2 时,$u \equiv v$ 的概率将变为 $O(1/n^2)$.

另一方面,注意到在稠密图中,顶点仍然具有公共的邻居.因此,如果从图 1(b)中删除边(v_5,v_6),(v_6,v_7)和(v_7,v_5),那么顶点 v_6 的父亲和儿子分别为$\{\{v_2,v_4\}\}$和$\{v_9\}$,顶点 v_7 的父亲和儿子分别为$\{\{v_2,v_4\}\}$和$\{v_9\}$,此时它们满足等价关系,可以归为一个等价类.于是可以得到一个新的压缩图 G_R^-,如图 1(c)所示.

基于上述发现,为了使 k-RPC 更好地适用于原始图较为稠密的情况,我们首先从原始图中删除一部分边使之变得稀疏,从而令更多的顶点可以满足等价关系.显然,删除的边集合 E^+将直接影响到 k-RPC 的压缩效果.于是,我们提出了最优边集问题来计算使压缩图最小的 E^+.下面将给出最优边集问题的形式化定义,并证明该问题是 NP-hard 的.

3.1 最优边集问题及其复杂度分析

记删除的边集合为 E^+,将删除了 E^+后的原始图 G 记为 $G^-=(V^-,E^-)=(V,E\backslash E^+)$.$G^-$可以通过 k-RPC 算法被压缩为 $G_R^- = (V_R^-, E_R^-)$. 可以发现,k-RPC 算法计算出的压缩图 G_R 是 $E^+=\varnothing$时的特例.因为删除边会使原始图中顶点间的最短距离增加,所以需要保存 E^+,故压缩图共包括两个部分:压缩图 G_R^- 本身和边集合 E^+.

定义 2. 给定原始图 G,$E^+ \subseteq E$ 是删除的边集合,$G^-=(V,E\backslash E^+)$可以通过 k-RPC 算法压缩为 G_R^-.最优边集问题就是计算出 E^+,使得$|G_R^-|+|E^+|$最小.

记 $OptG_R^-$ 为最优压缩图,那么计算 $OptG_R^-$ 是 NP-hard 的.我们通过将 3-SAT 问题规约到该问题来证明这一点.规约分为两个部分:1) 首先,将 3-SAT 问题规约为满足 $m>n$ 的 3-SAT 问题,其中,m 表示短语数,n 表示变量数;2) 然后,将满足 $m>n$ 的 3-SAT 问题规约为最优边集问题.对任意的 3-SAT 问题,我们将相应地构造一个原始图,并证明此 3-SAT 问题是可以满足的当且仅当该原始图压缩后不超过$(9m+2)n+59m+2n$.

定理 4. 给定任意的 3-SAT 问题 A,A 包含 n_A 个变量和 m_A 个短语,那么一定存在一个 3-SAT 问题 B,B 包含 n_B 个变量和 m_B 个短语,且 $m_B>n_B$,使得 A 可满足当且仅当 B 可满足.

证明:如果 $m_A>n_A$,令 $B=A$ 即可.否则,向 A 中加入 3 个变量 x_1,x_2,x_3 和 4 个短语$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3)$,得到一个新的 3-SAT 问题 A'.显然,A 可满足当且仅当 A'可满足.

重复上述过程(n_A-m_A+1)次,可以得到 B. □

定理 5. 给定任意的 3-SAT 问题满足 $m>n$(其中,n 表示变量数,m 表示短语数),均可将其规约为最优边集问题.

证明:给定一个 3-SAT 问题,其变量为 $x_1,x_2,\ldots,x_n$,短语为 $C_1,C_2,\ldots,C_m$,我们相应地构造一个原始图 G,如图 2 所示,具体构造过程如下:

- 首先,对于每个变量 x_i,$1 \leqslant i \leqslant n$,向 G 中加入顶点 x_i, $\bar{x}_i$ 和 y_i,如图 2 中第 3 行顶点所示;然后,再向 G 中加入 $2m$ 个顶点 $x_{i,1},x_{i,2},\ldots,x_{i,2m}$,如图 2 中第 2 行顶点所示,令 $x_{i,1},x_{i,2},\ldots,x_{i,m}$ 为顶点 x_i 的父亲,$x_{i,m+1}$, $x_{i,m+2},\ldots,x_{i,2m}$ 为顶点 $\bar{x}_i$ 的父亲,$x_{i,1},x_{i,2},\ldots,x_{i,2m}$ 为顶点 y_i 的父亲;最后,为每个 $x_{i,k}$ 加入 3 个顶点作为父亲,记作 $u_{i,k},v_{i,k}$ 和 $w_{i,k}$,其中,$1 \leqslant k \leqslant 2m$,如图 2 中第 1 行顶点所示;
- 然后,对于每个短语 C_j,$1 \leqslant j \leqslant m$,向 G 中加入顶点 $C_{j,1},C_{j,2}$ 和 $C_{j,3}$,如图 2 中第 4 行顶点所示;然后,再向 G 中加入 12 个顶点 $a_{j,1},a_{j,2},\ldots,a_{j,12}$,如图 2 中第 5 行顶点所示,令 $a_{j,1},a_{j,2},\ldots,a_{j,8}$ 为 $C_{j,1}$ 的儿子,$a_{j,5}$, $a_{j,6},\ldots,a_{j,12}$ 为 $C_{j,2}$ 的儿子,$a_{j,1},a_{j,2},\ldots,a_{j,4},a_{j,9},a_{j,10},\ldots,a_{j,12}$ 为 $C_{j,3}$ 的儿子;最后,为每个 $a_{j,r}$ 加入 3 个顶点作为儿子,记作 $b_{j,r},c_{j,r}$ 和 $d_{j,r}$,其中,$1 \leqslant r \leqslant 12$,如图 2 中第 6 行顶点所示;

- 最后,对于每个短语 $C_j=(z_1\vee z_2\vee z_3)$,向 G 中加入 3 条边$(z_1,C_{j,1})$,$(z_2,C_{j,2})$和$(z_3,C_{j,3})$.例如,如果 $C_j=(x_1\vee \bar{x}_2\vee x_3)$,那么加入的边为$(x_1,C_{j,1})$,$(\bar{x}_2,C_{j,2})$和$(x_3,C_{j,3})$,如图 2 中第 3 行与第 4 行顶点之间的边所示.至此,完成了对原始图 G 的构造.

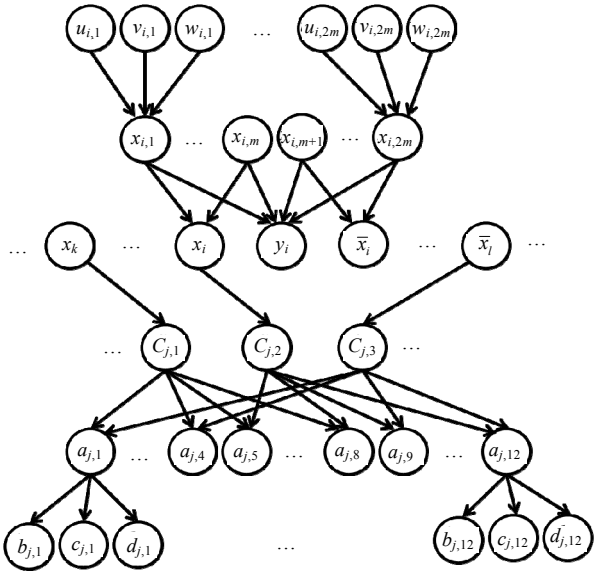


Fig.2 Instance for 3-SAT reduction

图 2 3-SAT 问题规约实例

下面考察如何压缩 G.可以发现,原始图 G 中的顶点分为 6 层且不同层次上的顶点没有相同的父亲或儿子,因此我们将逐层进行压缩.

首先,$u_{i,k}$,$v_{i,k}$和 $w_{i,k}$应该被压缩为 1 个顶点.否则,$x_{i,k}$一定和其他顶点具有相同的父亲或儿子,因此被压缩并删除了与 $u_{i,k}$,$v_{i,k}$和 $w_{i,k}$之间的边.故,共有以下两种压缩方案:1) 将 $x_{i,p}$和 $x_{i',q}$合并为一个顶点;2) 将 $x_{i,p}$和 $x_{i',q}$的父亲顶点分别压缩,其中,$1\leqslant i'\leqslant n$.当 $i=i'$时,对于任意的 $x_{i,p}$和 $x_{i,q}$,$1\leqslant p,q\leqslant 2m$,它们没有相同的父亲并且至多有两个相同的儿子,此时,显然后者的压缩效果更好;当 $i\neq i'$时,对于任意的 $x_{i,p}$和 $x_{i',q}$,$1\leqslant p,q\leqslant 2m$,它们没有相同的邻居,此时也是后者的压缩效果更好.类似地,$b_{j,r}$,$c_{j,r}$和 $d_{j,r}$应该被压缩为 1 个顶点,其中,$1\leqslant r\leqslant 12$,并且 $a_{j,r}$和 $a_{j',s}$不会合并,其中,$1\leqslant j'\leqslant n$,$1\leqslant s\leqslant 12$.

还可以发现,顶点 x_i和 y_i、$\bar{x}_i$和 y_i都有 m 个相同的父亲,且顶点 x_i和 $\bar{x}_i$的儿子都不超过 m 个,因此,应该将顶点 x_i和 y_i或者 $\bar{x}_i$和 y_i压缩为 1 个顶点.根据对称性,这两种压缩方案都可以采用.类似地,顶点 $C_{j,1}$,$C_{j,2}$和 $C_{j,3}$两两之间都有 4 个相同的儿子,且没有相同的父亲,因此应将其中两个压缩为 1 个顶点.同样根据对称性,这 3 种压缩方案都可以采用.

到目前为止,还有可能被压缩的顶点只剩下$\{x_i,\bar{x}_i\,|1\leqslant i\leqslant n\}$中的 n 个和$\{C_{j,1},C_{j,2},C_{j,3}|1\leqslant j\leqslant m\}$中的 m 个,且它们之间有 m 条边.记$\{x_i,\bar{x}_i\,|1\leqslant i\leqslant n\}$中未被压缩的顶点为$\{z_i|1\leqslant i\leqslant n\}$,$\{C_{j,1},C_{j,2},C_{j,3}|1\leqslant j\leqslant m\}$中未被压缩的顶点为$\{C_{j,f(j)}|1\leqslant j\leqslant m\}$,其中,$f$:$\{1,2,\ldots,m\}\to\{1,2,3\}$.可以发现:$\{z_i|1\leqslant i\leqslant n\}$中任意两个 z_i和 $z_{i'}$都没有相同的邻居,所以不会被合并;而$\{C_{j,f(j)}|1\leqslant j\leqslant m\}$中任意两个 $C_{j,f(j)}$和 $C_{j',f(j')}$也没有相同的儿子,如果它们没有相同的父亲,那么它们也不会被合并.

如果给定的 3-SAT 问题是可满足的,那么令$\{z_i|1\leqslant i\leqslant n\}$为一组可满足赋值.假设 $x_1=1$,$x_2=0$,…,$x_n=1$ 使 3-SAT 问题可满足,则令 $z_1=x_1$,$z_2=\bar{x}_2$,…,$z_n=x_n$.然后,通过调整映射 f,一定可以使每个 $C_{j,f(j)}$在$\{z_i|1\leqslant i\leqslant n\}$中都有一个父亲,而具有相同父亲的 $C_{j,f(j)}$可以被压缩为 1 个顶点,因此,$\{C_{j,f(j)}|1\leqslant j\leqslant m\}$会被压缩至不超过 n 个顶点;反之,如果$\{C_{j,f(j)}|1\leqslant j\leqslant m\}$可以被压缩至不超过 n 个顶点,那么$\{C_{j,f(j)}|1\leqslant j\leqslant m\}$的父亲顶点也一定不超过 n 个,而这些父亲

顶点对应的赋值也一定使 3-SAT 问题可满足.至此,我们完成了给定 3-SAT 问题到最优边集问题的规约.

下面分析压缩图的大小.首先,对于每一个变量,因为 $u_{i,k}$,$v_{i,k}$ 和 $w_{i,k}$ 被压缩为 1 个顶点,所以共有 $2m$ 个顶点;而 $x_{i,k}$ 未被压缩,所以仍有 $2m$ 个顶点;$u_{i,k}$,$v_{i,k}$,$w_{i,k}$ 与 $x_{i,k}$ 之间的边则被压缩为 $2m$ 条.假设 x_i 和 y_i 被合并,那么 $\{(x_{i,k},y_i)|m+1\leqslant i\leqslant 2m\}$ 将被删除并加入 E^+;$x_{i,k}$ 与 $\{x_i,y_i\}$ 之间的边被压缩为 m 条;$x_{i,k}$ 与 $\bar{x}_i$ 之间的边仍为 m 条.故,这一部分压缩图的大小为 $(9m+2)n$.

然后,对于每一个短语,类似地,它的儿子 $a_{j,r}$ 未被压缩,所以仍有 12 个顶点;而 $b_{j,r}$,$c_{j,r}$ 和 $d_{j,r}$ 被压缩为 1 个顶点,所以共有 12 个顶点;$a_{j,r}$ 与 $b_{j,r}$,$c_{j,r}$,$d_{j,r}$ 之间的边被压缩为 12 条.假设 $C_{j,1}$ 和 $C_{j,2}$ 被合并,那么 $\{(C_{j,1},a_{j,r})|1\leqslant r\leqslant 4\}$ 和 $\{(C_{j,2},a_{j,r})|9\leqslant r\leqslant 12\}$ 将被删除并加入 E^+;$\{(C_{j,3},a_{j,r})|1\leqslant r\leqslant 4\vee 9\leqslant r\leqslant 12\}$ 也会被删除并加入 E^+;$\{C_{j,1},C_{j,2}\}$ 与 $a_{j,r}$ 之间的边被压缩为 4 条;$C_{j,1}$,$C_{j,2}$ 与它们父亲之间的两条边也会被删除.再考虑到顶点 $\{C_{j,1},C_{j,2}\}$,那么,这一部分压缩图的大小为 $59m$.如果 3-SAT 问题可满足,那么 $\{C_{j,3}|1\leqslant j\leqslant m\}$ 可以被压缩至不超过 n 个顶点,$C_{j,3}$ 与 $\bar{x}_i$ 之间的边也不超过 n 条.因此,压缩图的规模将不超过 $(9m+2)n+59m+2n$. □

显然,定理 4 中的规约和定理 5 中原始图的构造都可以在多项式时间内完成,因此,最优边集问题是 NP-hard 的.

3.2 图压缩算法 k-GRPC

我们给出了一种启发式图压缩算法 k-GRPC,如算法 2 所示.k-GRPC 算法迭代地从原始图 G 中选择“合适的”两个顶点进行合并,直至所有的顶点均不能再被合并.压缩图随着算法的运行不断发生变化,其最终结果为压缩图 G_R^-.

Algorithm 2. k-GRPC.

Input: Compressed Graph $G_R=(V_R,E_R)$;

Output: Compressed Graph G_R^-.

① $V_{tmp}=V_R$, $E_{tmp}=E_R$, $count=0$;
② **while** $count<s$ **do**
//连续采样得到将要压缩的顶点对
③ random generate u;
④ pick w from u's neighbors randomly;
⑤ choose v from w's neighbors randomly;
⑥ compute $C(u,v)$, $Del(u,v)$, $Exr(u,v)$;
⑦ **If** $|C(u,v)|>Del(u,v)-|Exr(u,v)|$ **do**
⑧ compress u and v;
⑨ $count=0$, update V_{tmp} and E_{tmp} accordingly;
⑩ **else**
⑪ $count$++;
⑫ **end if**
⑬ **end while**
⑭ **Return** $G_R^-=(V_{tmp},E_{tmp})$;

判断图中两个顶点 u 和 v 是否“合适”,取决于将它们归为一个等价类后是否可以提高压缩比.如算法第 7 行所示(关于第 7 行中的判断条件,我们将稍后加以说明):如果图中的两个顶点在压缩后可以使压缩图的大小 $|G_R^-|+|E^+|$ 减少,则它们是“合适的”;如果图中任意两个顶点都不是“合适的”,则算法终止.判断两个顶点是否“合适”,应考察两个因素:一方面,每合并两个顶点都会使 $|V_R^-|$ 减小 1,故需要考察 $|E_R^-|+|E^+|$ 的变化;另一方面,还需要考察 $|E^+|$ 的变化,这是由于 $|E^+|$ 中的边没有被压缩,故,$|E^+|$ 的增大会导致压缩比的下降.

综上所述,在选择“合适的”顶点对 u 和 v 进行合并时,算法首先使 $|E^+|$ 增加的最少;当出现相等的情况时,再使

$|E_R^-|+|E^+|$下降的最多.

由于 k-RPC 算法是将满足等价关系的顶点合并为一个等价类,而满足等价关系的顶点在合并后不会从原始图 G 中删除边,即,不会改变$|E^+|$,因此,如果在原始图上调用 k-GRPC 算法,当$|E^+|$第 1 次不为 0 时,计算出的压缩图恰为 G_R.因此,在使用 k-GRPC 算法时可以直接以压缩图 G_R 作为输入.值得注意的是,k-GRPC 算法可能进一步压缩 G_R 中的单个顶点或者压缩后的顶点.

例如,在对图 1(a)中的原始图 G 进行压缩时,将直接使用图 1(b)中的压缩图 G_R 作为输入.因为顶点 v_6 的父亲和儿子分别为$\{\{v_2,v_4\},v_5\}$和$\{v_7,v_9\}$,顶点 v_7 的父亲和儿子分别为$\{\{v_2,v_4\},v_6\}$和$\{v_5,v_9\}$,所以合并顶点 v_6 和 v_7 需要删除边(v_5,v_6),(v_6,v_7)和(v_7,v_5),此时,$|E^+|$会增加 3;而它们相同的父亲和儿子包括$\{v_2,v_4\}$和 v_9,故,$|E_R^-|+|E^+|$会减小 2.因此,v_6 和 v_7 是“合适的”,故算法将合并顶点 v_6 和 v_7 完成 1 次压缩.此后,图中任意两个顶点均不能合并,算法终止,最终得到了图 1(c)所示的压缩图 G_R^-.

下面来逐步说明算法第 7 行的判断条件.

首先,在算法执行过程中,需要为每一对顶点计算它们相同的父亲和儿子的 $C(u,v)$以及不同的父亲和儿子的 $Ex(u,v)$,其中,

$$C(u,v)=(child(u)\cap child(v))\cup(father(u)\cap father(v)) \tag{1}$$

$$Ex(u,v)=(child(u)\cup child(v)\cup father(u)\cup father(v))-C(u,v) \tag{2}$$

在压缩两个顶点 u 和 v 之前,为了使它们具有相同的父亲和儿子,需要从图中删除顶点 u 和 v 与它们不同的父亲和儿子 $Ex(u,v)$之间所有的边.因为顶点 u 和 v 之间的边(u,v)在 $child(u)$和 $father(v)$中重复计算了 2 次,所以,实际需要考虑的不同父亲和儿子 $Exr(u,v)$满足:

$$Exr(u,v)=Ex(u,v)-E(u,v) \tag{1}$$

其中,$E(u,v)$满足:

$$E(u,v)=\begin{cases}u, & (u,v)\in E,(v,u)\notin E\vee(v,u)\in E,(u,v)\notin E\\ \varnothing, & \text{otherwise}\end{cases} \tag{4}$$

以对于图 1(b)中的压缩图 G_R 为例,根据公式(3)和公式(4),我们有$|C(v_6,v_7)|=2$,$|Exr(v_6,v_7)|=3$.u 和 v 与 $Exr(u,v)$之间边的数量并不一定恰好为$|Exr(u,v)|$,因为顶点 u,v 和 $Exr(u,v)$中的顶点可能是等价类,所以,删除的边数 $Del(u,v)$满足:

$$Del(u,v)=\sum_{w\in Exr(u,v)\wedge w\in deg(u)}|[u]_R|\cdot|[w]_R|+\sum_{w\in Exr(u,v)\wedge w\in deg(v)}|[v]_R|\cdot|[w]_R| \tag{5}$$

根据公式(5),我们有 $Del(v_6,v_7)=2+1=3$.

如果顶点 u 和 v 是等价类,那么删除它们与 $Exr(u,v)$之间的边就会使已经压缩的边被解压缩.因为合并顶点 u 和 v 需要删除 $Del(u,v)$条边,而在合并前的图中,这部分边压缩为$|Exr(u,v)|$条,所以合并 u 和 v 后压缩图的规模将增加 $Del(u,v)-|Exr(u,v)|$;如果合并的收益$|C(u,v)|<Del(u,v)-|Exr(u,v)|$,那么顶点 u 和 v 不会被合并,如算法第 7 行所示.

由于选择最优的顶点对需要一一考察 $O(|V_R|^2)$个顶点对,对于规模较大的图数据,上述时间开销将非常巨大.本文给出了线性时间的随机采样策略,如算法第 2 行~第 5 行所示.采样过程首先从 V_R 中随机选择一个顶点 u,然后从 u 的邻居中随机选择顶点 w,再从 w 的邻居中随机选择顶点 v,最后判断顶点 u 和 v 是否是“合适的”.这样做是因为“合适的”顶点对至少应有一个公共邻居.如果连续 s 次随机采样到的顶点对都不是“合适的”,则终止采样算法.易知,该采样服从几何分布.假设每一次采样成功的概率为 p,那么不超过 s 次得到“合适”顶点对的概率为 $1-(1-p)^s$.为了使 s 次采样成功的概率不低于 p_0,可以令 $s=\lceil\log_{(1-p)}(1-p_0)\rceil$.关于 p 值的计算我们仍然可以通过采样方法来估计,采样策略与算法 2 第 3 行~第 5 行的类似,假设在总共 N 次采样中得到了 M 对“合适的”顶点,那么 $p=M/N$,本文的实验中设定 $p_0=0.999$,$N=10000$.尽管每压缩一对“合适”顶点就会使剩余的“合适”顶点对减少,但是因为此时顶点数也会随之减小,故在 s 次连续采样过程中将不更新 p 值.

下面分析 k-GRPC 算法的近似比.由于 k-GRPC 算法是以 G_R 作为输入,故$|G_R^-|+|E^+|\leqslant|G_R|$.对于最优压缩图

$OptG_R^-$,可以证明$|OptG_R^-|+|E^*|\geqslant|V_R|$,其中,$E^*$表示最优边集问题的最优解.于是,$k$-GRPC 算法的近似比就是$(1+|E_R|/|V_R|)$.

定理 6. 给定原始图 G,$G_R=(V_R,E_R)$是 k-RPC 算法得到的压缩图,E^*和 $OptG_R^-$ 是最优边集问题的最优解和最优压缩图,那么$|OptG_R^-|+|E^*|\geqslant|V_R|$.

证明:易知,$OptG_R^-$ 可以在 G_R 上通过删除边和划分等价类得到.这就意味着,对于任意的顶点 u 和 v,如果$[u]_R=[v]_R$,那么它们在 $OptG_R^-$ 中也一定被划分为一个等价类.否则,记 u 和 v 所属等价类为$[u]_{Opt}$ 和$[v]_{Opt}$,用$nei([u]_{Opt})$表示$[u]_{Opt}$ 在压缩图中的邻居,用 $neiE^*(u)$表示 u 在 E^*中的邻居.不妨假设$|nei([u]_{Opt})|+|neiE^*(u)|\leqslant|nei([v]_{Opt})|+|neiE^*(v)|$.因为 $father(u)=father(v)$且 $child(u)=child(v)$,因此可将 v 移动至等价类$[u]_{Opt}$.显然,压缩图的规模不会扩大.

因此,压缩图 $OptG_R^-$ 是将 G_R 中的等价类进一步合并了.因为任意两个等价类的合并都至少向 E^*中增加 1 条边,所以$|V_{Opt}|+|E^*|\geqslant|V_R|$.那么显然,$|OptG_R^-|+|E^*|\geqslant|V_R|$. □

事实上,尽管压缩图 G_R^- 中的顶点代表一个等价类,而边集合 E^+中的顶点表示单个顶点,我们仍然可以将二者存储在一起.为保证查询的正确与效率,存储应当满足以下规则:1) 对压缩图 G_R^- 与 E^+中的边,仍然使用邻接表的形式存储,且保证有序;2) E^+中的顶点使用原始图中的 id,G_R^- 中的顶点使用不同的 id 以示区分;3) 压缩图 G_R^- 与 E^+中的边交替存储,首先是 V_R^- 中第 1 个顶点$[u_1]_R$ 在 E_R^- 中的邻接表,然后是$[u_1]_R$ 中每一个顶点在 E^+中的邻接表,再然后是 V_R^- 中第 2 个顶点$[u_2]_R$ 的邻接表,然后又是$[u_1]_R$ 中每一个顶点在 E^+中的邻接表,….例如,对于图 1(b)所示的压缩图 G_R,通过向 E^+中增加(v_5,v_6),(v_6,v_7)和(v_7,v_5),可以得到图 1(c)所示的压缩图 G_R^-,那么根据规则 2,$\{v_1\}$的 id 为 10,$\{v_2,v_4\}$的 id 为 11,以此类推;然后根据规则 3,边将按照以下顺序存储:

$$(v_{10},v_{11}),(v_{10},v_{12}),\ldots,(v_{13},v_{15}),(v_5,v_6),(v_{14},v_{16}),(v_6,v_7),(v_7,v_5),$$

其中,v_{13},v_{14},v_{15} 和 v_{16} 分别表示$\{v_5\}$,$\{v_6,v_7\}$,$\{v_8\}$和$\{v_9\}$.将压缩图 G_R^- 和边集合 E^+连续存放的好处是:可以连续访问一个等价类中全部顶点的邻居,从而减少 I/O 代价.

下面分析算法 2 的时间复杂度.在每次迭代过程中,k-GRPC 将两个顶点压缩为 1 个,故算法迭代的次数为$O(s|V_R|)$.在每次迭代中,算法第 3 行~第 5 行的随机采样过程将从随机生成顶点 u 的一个随机邻居中再次随机选择一个邻居,故,时间复杂度为 $O(avg(deg(u)))$;算法第 6 行中,计算 $C(u,v)$,$Exr(u,v)$和 $Del(u,v)$需要顺序访问顶点 u 和 v 的邻居,其时间复杂度为 $O(deg(u)+deg(v))=O(avg(deg(u)))=O(|E_R|/|V_R|)$;算法第 8 行~第 9 行中,合并两个顶点并更新压缩图可以在常数时间内完成.综上所述,算法 2 的时间复杂度为 $O(s|E_R|)$.另外,算法 2 使用了 k-RPC 算法计算的压缩图作为输入,故 k-GRPC 算法总的时间复杂度为 $O(s|E_R|+|E|\log|V|)$.

当原始图发生了动态更新时,可以通过如下操作实现更新:1) 对于孤立顶点的插入/删除,可以直接向孤立顶点等价类中插入或者从其所在等价类中删除即可;2) 对于边的插入,可以直接将新增的边加入 E^+;3) 对于边的删除,可以从压缩图 G_R^- 中删除边$([u]_R,[v]_R)$,其中,$[u]_R$ 和$[v]_R$ 表示顶点 u 和 v 所属的等价类.另外,还应把$[u]_R$中所有顶点与$[v]_R$中所有顶点之间的边加入 E^+.易知,边的插入/删除会导致 E^+不断增大并影响查询效率,而由于压缩图 G_R 可以随时准确更新,我们可以定期地在 G_R 上调用 k-GRPC 算法重新计算压缩图 G_R^- 和 E^+.由于动态图并不是本文的研究对象,故对压缩图 G_R 和 G_R^- 上的动态更新技术本文不再作详细探讨.

3.3 基于k-GRPC压缩图的查询处理算法

k-GRPC 算法虽然可以提高压缩比,却不能直接在压缩图 G_R^- 上完成查询处理.这是因为,当使用压缩图 G_R 计算 $d(G,u,v)$时,根据定理 1,可以直接计算出 $d(G_R,[u]_R,[v]_R)$即可;而由于原始图 G 中一部分边被删除并保存在 E^+中,使用压缩图 G_R^- 计算的 $d(G_R^-,[u]_R,[v]_R)$是 $d(G,u,v)$的上界.

我们仍然可以使用类似广度优先的算法完成查询,见算法 3.算法需要建立一个优先队列 QA,队列中的顶点可以是单个顶点或者等价类,并仍然按照与顶点 u 之间的距离从小到大排序.对于队首顶点 x,其邻居一定

包含两部分:压缩图 G_R^- 中的等价类和 E^+中的顶点,分别如算法第 6 行~第 8 行和第 9 行~第 11 行所示.下面分别讨论 x 是单个顶点和等价类的情况:

1) 当 x 是单个顶点时, G_R^- 中的邻居可以从$[x]_R$的邻接表访问,E^+中的邻居可以直接从 x 在 E^+中的邻接表访问;
2) 当 x 是等价类时, G_R^- 中的邻居可以直接从 x 的邻接表访问,E^+中的邻居可以从 x 的所有顶点在 E^+中的邻接表访问.

由于上述两种情况比较相似,在算法 3 中不再加以区分.

Algorithm 3. *QueryG*$^-$.

Input: Compressed Graph G_R^-, Removed Edge Set E^+, Vertex u and v, Value k;

Output: 'Yes' if u can reach v within k-hops or 'No' otherwise.

① $d[u,u]=0$, *push*(u,Q);
② **for** each $v\in V_R^-$ **do**
③ $d[u,v]=\infty$;
④ **while** Q is not empty **do**
⑤ w=*pop*(Q);
⑥ **For** each vertex $x\in V_R^- \wedge (w,x)\in E_R^- \wedge d[u,x]>d[u,w]+1$
//访问压缩图中的边
⑦ $d[u,x]=d[u,w]+1$, *pushQ*(x,Q);
⑧ **end for**
⑨ **For** each vertex $x\in[w]_R \wedge (x,y)\in E^+ \wedge d[u,y]>d[u,x]+1$
//访问 E^+中的边
⑩ $d[u,y]=d[u,x]+1$, *pushQ*(y,Q), split y from $[y]_R$;
⑪ **end for**
⑫ **end while**
⑬ **If** $d[u,[v]_R]\leqslant k$
⑭ **Return** 'Yes';
⑮ **Return** 'No';

需要指出的是:当单个顶点 x 出现在队首时,一定满足 $d(u,x)<d(u,[x]_R)$.这里,$[x]_R$ 是顶点 x 在 G_R^- 中所属的等价类.为了区分顶点 x 和$[x]_R$中的其他顶点,算法将 x 从等价类$[x]_R$中分裂并形成一个新的等价类.新等价类仅包含 x 一个顶点.

下面分析查询算法 3 的复杂度:一方面,优先队列 QA 中出现的顶点包括 G_R^- 中的等价类和这些等价类分裂出的顶点,而由于每访问 E^+中的一条边至多将导致一个顶点的分裂,故,QA 中的顶点不超过 $O(|V_R^-|+|E^+|)$个;另一方面,算法访问的边集为 E_R^- 和 E^+.因此,在压缩图 G_R^- 上查询处理的时间复杂度为 $O(|V_R^-|+|E_R^-|+|E^+|)$.

因为在压缩图 G_R 上查询的时间复杂度为 $O(|V_R|+|E_R|)$,所以使用压缩图 G_R^- 查询的时间复杂度更低.但是,因为在查询压缩图 G_R 时只需要访问每个顶点和边各 1 次,而算法 3 不仅需要访问 E^+中的边 1 次,并且由于 E^+中边的访问,又可能导致等价类分裂 $O(|E^+|)$次,即,需要额外访问 $O(|E^+|)$个顶点.基于上述原因,通过真实数据上的实验验证:

- 在稀疏图上,两种压缩算法的压缩比相当.此时,基于 k-RPC 压缩算法的查询处理效率略好于基于 k-GRPC 压缩算法的查询处理效率;
- 在稠密图上,由于 k-GRPC 具有较好的压缩比,所以其查询效率明显高于基于 k-RPC 算法的查询效率.

4 实验结果

在实验部分,我们在真实数据上对两种压缩算法的有效性、压缩效率和查询效率进行了考察.对比实验包括基于顶点覆盖的索引 VC[1],2-hop VC[1]和 BFS 算法.下面首先介绍实验考察的几种测度,包括压缩算法压缩比、运行时间和查询处理时间;随后介绍使用的实验数据、实验环境和对比实验;最后给出实验的对比及分析.

在考察算法的压缩比时,分别用 CR_k 和 CR_{kG} 表示压缩方法 k-RPC 和 k-GRPC 算法的压缩比,其中,

$$CR_k=(|V_R|+|E_R|)/(|V|+|E|),CR_{kG}=(|V_R^-|+|E_R^-|+|E^+|)/(|V|+|E|).$$

可知:压缩比越小,压缩图的规模也越小,说明压缩算法更加有效;在考察压缩算法的运行时间时,由于 k-GRPC 算法是以 k-RPC 算法计算的压缩图作为输入,因此我们认为,k-GRPC 算法的压缩时间应包含 k-RPC 的压缩时间;在考察查询处理时间时,对比了两种基于压缩图的查询处理算法的时间开销与在原始图上直接进行查询处理的时间开销.

本文的全部实验在真实的数据集上进行.实验数据包括:1) 斯坦福大学的共享数据集(http://snap.stanford.edu/data/#twitter);2) 国内科研数据共享网站数据堂中数据挖掘数据库(http://www.datatang.com/).本文使用的数据规模在 685K~8.3M 之间,包括了以下网络:1) Email-EuAll[1]、欧盟科研机构之间的邮件网络和 wiki-Talk[1]、维基百科网站上的交互网络;2) Web-NotreDame[1],Web-Google[1],Web-Stanford[1] 和 Web-BerkStan[1] 等网络图;3) Twitter[2],一个在线社交网站.表 1 中给出了本文使用的社交网络的特性,其中,p 表示在压缩图 G_R 上随机采样到“合适”顶点对的概率,s 表示为了使 s 次采样成功的概率不低于 99.9%的采样次数.

Table 1 Properties for real datasets

表 1 实验中使用数据集的基本特性

数据集	顶点数	边数	平均度数	p	s
Email-EuAll	265 214	420 045	3.17	0.011	625
twitter	465 023	835 541	3.59	0.024	285
wiki-Talk	2 394 385	5 021 410	4.20	0.036	189
Web-NotreDame	325 729	1 497 134	9.20	0.179	35
Web-Google	875 713	5 105 039	11.7	0.254	24
Web-Stanford	281 903	2 312 497	16.4	0.311	19
Web-BerkStan	685 230	7 600 595	22.2	0.329	18

上述实验使用 C++实现,在双核 3.40GHz Intel Core(PM) D CPU,32.0GB 内存,Windows Vista 系统的台式机上运行.

基于顶点覆盖的索引 VC 和 2-hop VC 是目前能够有效处理 k 可达查询的方法,其索引的规模甚至不超过某些支持可达查询索引的规模[1].但是这两种索引仍然不能扩展到规模较大的图上,且只能回答特定 k 值下的 k 可达查询,而如果需要精确查询,则必须构建一系列索引.由于 2-hop VC 要求 $k>2h=4$,故,在实验中构造了支持 k 值为 5 的可达查询索引.

表 2 给出了两种压缩方法和 VC,2-hop VC 在全部数据集上的压缩比对比情况.按照图数据的平均度数,我们对 7 个真实数据集上的压缩结果进行了比较.可以发现:在最稀疏的 Email-EuAll 上,k-RPC 算法的压缩比最好,可以达到 35.2%,但是随着图的平均度数的增大,k-RPC 算法的压缩比也随之上升,其中在 Web-Google 上,压缩比最差,达到了的 81.1%.注意到,k-RPC 在平均度数更大的 Web-Stanford 和 Web-BerkStan 上的压缩比反而更好,处于 70%~75%左右.这是由于在校园社会网络中更多出现相同邻居的特点,即,一个学术团队内的人往往有着共同的联络对象.但从整体上来看,随着图平均度数的增大,k-RPC 的压缩效果会随之下降.

我们还可以发现,k-GRPC 的压缩比一直好于 k-RPC 的压缩比.这是由于 k-GRPC 算法是在 k-RPC 的基础上进一步压缩的.当原始图比较稀疏时,例如在 wiki-Talk 上,两种算法的压缩比非常接近,分别为 49.4%和 49.1%;而随着图的平均度数的增加,k-GRPC 的压缩比将明显好于 k-RPC 的压缩比.例如在 Web-Stanford 上,两种算法的压缩比分别为 75.1%和 62.1%.这是由于 k-RPC 严格的等价关系在较为稠密的图上不易满足所致.

VC 和 2-hop VC 构建的索引要远远大于原始图的规模,且随着图平均度数的增大显著递增.在 Web-Google

和 Web-BerkStan 上,两种索引图甚至分别超过了 700M 和 2G 条边,并超过了原始图规模的 119 倍和 241 倍.如果进一步考虑到索引需要保存的权值信息,上述索引将更加庞大.而如此庞大的索引却仅能准确回答 k 值为 5 的 k 可达查询,而随着 k 值的增大,基于索引的方法还需要构建更大规模的索引来支持查询处理.

Table 2 Compressed graph size of k-RPC, k-GRPC and index size of VC, 2-hop VC

表 2 k-RPC,k-GRPC 的压缩图大小与 VC,2-hop VC 的索引大小

	数据集	Email-EuAll	twitter	wiki-Talk	Web-NotreDame	Web-Google	Web-Stanford	Web-BerkStan
k-RPC	顶点数	60 343	100 591	573 411	148 319	638 888	217 000	493 161
	边数	180 669	452 787	3 085 885	1 223 316	4 246 521	1 730 543	5 328 042
	压缩比(%)	35.2	42.5	49.4	75.2	81.7	75.1	70.3
k-GRPC	顶点数	58 425	82 496	570 966	140 066	571 688	173 686	473 127
	边数	142 175	231 843	2 904 497	1 071 849	2 850 497	809 759	4 568 367
	$\|E^+\|$	35 425	200 488	177 086	115 165	946 498	564 586	231 435
	压缩比(%)	34.4	40.0	49.1	72.8	73.0	60.0	63.6
VC	顶点数	36 359	4 992	152 686	122 231	539 319	201 538	457 928
	边数	9 400 171	4 037 161	>1G	121 039 294	>841M	>514M	>2G
	压缩比(倍)	13.77	3.11	>134	66.47	>140	>198	>241
2-hop VC	顶点数	2 734	2 515	101 826	96 105	426 810	183 403	409 145
	边数	4 574 446	1 657 599	>712M	100 951 906	>716M	>402M	>2G
	压缩比(倍)	6.68	1.28	>96	55.43	>119	>155	>241

表 3 给出了两种压缩算法的压缩时间与 VC,2-hop VC 方法构建索引时间的对比.可以发现:k-RPC 和 k-GRPC 算法的压缩时间是随着数据集规模的扩大呈线性增加的,这与之前的分析是一致的;而构建 VC,2-hop VC 索引的时间则是随着数据集规模的扩大而显著增加的,这是由于两种方法构建索引规模的显著扩大所致.其中,在规模为 2.59M 的 Web-Stanford 上,VC 和 2-hop VC 甚至不能在 1 小时内完成索引的构造.

Table 3 Compressing time of k-RPC, k-GRPC and index construction time of VC, 2-hop VC

表 3 k-RPC,k-GRPC 的压缩时间和 VC,2-hop VC 的索引建立时间

	压缩时间(s)			
数据集	k-RPC	k-GRPC	VC	2-hop VC
Email-EuAll	2.6	4.7	89.1	22.9
twitter	1.3	7.6	15.8	32.5
wiki-Talk	66.5	77.5	>3600	>3600
Web-NotreDame	87.6	102.4	>3600	>3600
Web-Google	92.7	113.3	>3600	>3600
Web-Stanford	37.9	75.9	>3600	>3600
Web-BerkStan	162.3	206.5	>3600	>3600

表 4 给出了基于压缩算法的查询时间与基于索引方法的查询时间对比情况,实验随机生成了 10 000 组查询,并给出了平均查询时间.可以发现:当原始图比较稀疏时,基于 k-RPC 算法的查询效率明显好于 BFS 算法,例如在 Email-EuAll,twitter 上,前者的查询效率是后者的 2.5 倍以上;而随着图的平均度数变大,基于 k-RPC 的查询效率将逐渐接近原始图上的查询效率,达到 1.3 倍左右.这是由于,基于压缩图的查询效率是与压缩图的规模相关的,随着图数据变得稠密,k-RPC 的压缩效果会下降,故查询效率也会随之下降.

我们还可以发现,基于 k-GRPC 的查询效率在稀疏图上略低于基于 k-RPC 的查询效率.这是由于两种压缩算法在稀疏图上的压缩比非常接近,而基于 k-GRPC 的查询由于需要访问 $|V_R^-|+|E_R^-|+2|E^+|$个顶点和边,故,查询效率会有所下降.当图变得比较稠密时,基于 k-GRPC 的查询效率会明显好于基于 k-RPC 的查询处理.例如,在 Web-Stanford 上,两种基于压缩图的查询处理算法效率分别是 BFS 算法的 1.35 倍和 1.50 倍.这是由于,在比较稠密的图上,k-GRPC 仍然可以对 k-RPC 计算的压缩图进行有效的压缩,从而提高查询效率.

基于两种索引方法的查询效率是与索引规模直接相关的,故,在规模较小的 Email-EuAll 上,其查询效率较好,达到了 BFS 算法的 750 倍左右;但是随着数据规模的扩大,在规模约为 Email-EuAll 1.9 倍的 twitter 上,两种查询算法的效率迅速下降至仅为 BFS 算法的 150 倍;而当数据集进一步扩大时,基于索引的方法将由于索引过

于庞大而无法使用.尽管在数据集规模较小时,基于索引的查询更加有效,但是由于它们不具备较好的可扩展性,因此无法适用在规模更大的数据集上.

Table 4 Query performance of k-RPC, k-GRPC, VC, 2-hop VC and BFS

表 4 k-RPC,k-GRPC,VC,2-hop VC 和 BFS 的查询效率

	查询时间(ms)				
数据集	k-RPC	k-GRPC	BFS	VC	2-hop VC
Email-EuAll	0.87	1.09	2.16	0.009 4	0.002 9
twitter	3.05	3.08	7.81	0.095	0.047
wiki-Talk	110.86	111.05	240.69	–	–
Web-NotreDame	13.73	13.50	18.33	–	–
Web-Google	123.21	118.43	139.31	–	–
Web-Stanford	45.00	40.33	52.94	–	–
Web-BerkStan	137.74	120.84	173.58	–	–

综上所述,基于索引的方法只能适用于数据集规模较小的情况.对于规模较大的数据集,当原始图比较稀疏时,两种压缩算法的压缩比较为接近,此时,k-RPC 算法更为适用;当原始图比较稠密时,k-GRPC 算法的压缩比会明显好于 k-RPC 的压缩比,并具有更好的查询效率,故,此时 k-GRPC 更加适用.

5 结束语

本文研究了基于图压缩技术的 k 可达查询处理算法,提出了图压缩算法 k-RPC 及在压缩图上无需解压缩的查询处理算法,证明了基于图压缩算法 k-RPC 的查询处理的正确性,并证明了 k-RPC 在所有支持 k 可达查询的基于等价类的图压缩算法中是最优的.考虑到 k-RPC 的等价关系比较严格,在稠密图上不易满足,本文允许从原始图中删除部分边,并提出了最优边集问题,且证明了该问题是 NP-hard 的;还给出了线性的近似图压缩算法 k-GRPC 及在压缩图上无需解压缩的查询处理算法.通过真实数据上的实验结果表明:两种压缩算法的压缩比在稀疏图上可以达到 45%,在稠密图上分别可以达到 75%和 67%,而查询效率在稀疏图上可以提高 2.5 倍.在今后的工作中,我们将进一步探讨针对本文压缩方法的有效查询处理算法和外存上有向图的图压缩算法.

致谢 在此,我们向对本文提出宝贵审稿建议的审稿专家以及哈尔滨工业大学计算机科学与技术学院的李建中教授表示衷心的感谢.

References:

[1] Cheng J, Shang Z, Cheng H, Wang H, Yu JX. *K*-Reach: Who is in your small world? In: Proc. of the 2012 VLDB Endowment. Istanbul: ACM Press, 2012. 1292−1303. http://research.microsoft.com/pubs/166387/vldb2012.pdf

[2] Jin R, Ruan R, Dey S, Yu JX. Scarab: Scaling reachability computation on large graphs. In: Proc. of the 2012 Int'l Conf. on Management of Data. Scottsdale: ACM Press, 2012. 169−180. [doi: 10.1145/2213836.2213856]

[3] Jin RM, Xiang Y, Ruan N, Fuhry D. 3-Hop: A high-compression indexing scheme for reachability query. In: Proc. of the 35th SIGMOD Int'l Conf. on Management of Data. Providence: ACM Press, 2009. 813−826. [doi: 10.1145/1559845.1559930]

[4] Jin RM, Xiang Y, Ruan N, Wang HX. Efficiently answering reachability queries on very large directed graphs. In: Proc. of the 2008 Int'l Conf. on Management of Data. Vancouver: ACM Press, 2008. 595−608. [doi: 10.1145/1376616.1376677]

[5] Yu JX, Cheng J. Graph reachability queries: A survey. In: Managing and Mining Graph Data. New York: Springer-Verlag, 2010. 181−215.

[6] Cohen E, Halperin E, Kaplan H, Zwick U. Reachability and distance queries via 2-hop labels. SIAM Journal on Computing, 2003, 32(5):1338−1355. [doi: 10.1137/S0097539702403098]

[7] Wei F. TEDI: Efficient shortest path query answering on graphs. In: Proc. of the 2010 Int'l Conf. on Management of Data. Indianapolis: ACM Press, 2010. 99−110. [doi: 10.1145/1807167.1807181]

[8] Cheng J, Yu JX. On-Line exact shortest distance query processing. In: Proc. of the 12th Int'l Conf. on Extending Database Technology. Saint Petersburg: ACM Press, 2009. 481−492. [doi: 10.1145/1516360.1516417]

[9] Xiao YH, Wu WT, Pei J, Wang W, He ZY. Efficiently indexing shortest paths by exploiting symmetry in graphs. In: Proc. of the 12th Int'l Conf. on Extending Database Technology. Saint Petersburg: ACM Press, 2009. 493−504. [doi: 10.1145/1516360.1516418]

[10] Fan WF, Li JZ, Wang X, Wu YH. Query preserving graph compression. In: Proc. of the 2012 Int'l Conf. on Management of Data. Scottsdale: ACM Press, 2012. 157−168. [doi: 10.1145/2213836.2213855]

[11] Aho AV, Garey MR, Ullman JD. The transitive reduction of a directed graph. SIAM Journal on Computing, 1972,1(2):131−137. [doi: 10.1137/0201008]

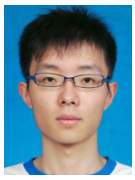
[12] Simon K. An improved algorithm for transitive closure on acyclic digraphs. Theoretical Computer Science, 1988,58(1):325−346. [doi: 10.1016/0304-3975(88)90032-1]

[13] Jagadish HV. A compression technique to materialize transitive closure. ACM Trans. on Database System, 1990,15(4):558−598. [doi: 10.1145/99935.99944]

[14] Cheng J, Ke YP, Chu S, Cheng C. Efficient processing of distance queries in large graphs: a vertex cover approach. In: Proc. of the 2012 Int'l Conf. on Management of Data. Scottsdale: ACM Press, 2012. 457−468. [doi: 10.1145/2213836.2213888]

[15] Cheng J, Zhu LH, Ke YP, Chu S. Fast algorithms for maximal clique enumeration with limited memory. In: Proc. of the 18th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. Beijing: ACM Press, 2012. 1240−1248. [doi: 10.1145/2339530.2339724]

[16] Schank T, Wagner D. Finding, counting and listing all triangles in large graphs, an experimental study. In: Proc. of the 4th Int'l Workshop on Efficient and Experimental Algorithms. Santorini Island: Springer-Verlag, 2005. 606−609. [doi: 10.1007/11427186_54]



李鸣鹏(1989－),男,黑龙江勃利人,博士生,主要研究领域为图数据的查询处理.
E-mail: lmp@hit.edu.cn

高宏(1966－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据挖掘,无线传感器网络.
E-mail: honggao@hit.edu.cn

邹兆年(1979－),男,博士,讲师,CCF 会员,主要研究领域为图数据挖掘.
E-mail: znzou@hit.edu.cn