软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(9):2265−2277 [doi: 10.13328/j.cnki.jos.005044]

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

基于图压缩的最大 Steiner 连通 *k* 核查询处理*

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最大 Steiner 连通 *k* 核查询处理,提出了一种支持最大 Steiner 连通 *k* 核查询的图压缩算法 SC,证明了基于 SC 压缩算法的查询正确性.由于最大 Steiner 连通 *k* 核查询仅需要找到符合要求的连通区域,提出了图压缩算法 TC,进一步将压缩图压缩为树.证明了基于压缩树的查询正确性,并提出了线性时间的无需解压缩的查询处理算法.真实和虚拟数据上的实验结果表明:压缩算法平均可将原始图压缩掉 88%,且对于稠密的原始图,压缩算法的压缩效果更好,可将原始图压缩掉 90%,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算法效率更好,平均提升了 1~2 个数量级.

关键词: 最大 Steiner 连通 *k* 核;图压缩;等价类;查询处理;压缩比

中图法分类号: TP311

中文引用格式: 李鸣鹏,高宏,邹兆年.基于图压缩的最大 Steiner 连通 *k* 核查询处理.软件学报,2016,27(9):2265−2277. http://www.jos.org.cn/1000-9825/5044.htm

英文引用格式: Li MP, Gao H, Zou ZN. Maximum Steiner connected *k*-core query processing based on graph compression. Ruan Jian Xue Bao/Journal of Software, 2016,27(9):2265−2277 (in Chinese). http://www.jos.org.cn/1000-9825/5044.htm

Maximum Steiner Connected *k*-Core 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)

Abstract: This paper focuses on maximum Steiner connected *k*-core query processing based on graph compression, and proposes a maximum Steiner connected *k*-core query preserving graph compression algorithm, SC. The correctness of querying based on SC algorithm is proved. Since maximum Steiner connected *k*-core query only requires a connected component which satisfies certain properties, graph compression algorithm TC is proposed to further compact the compressed graph into a tree. It is proved that querying based on the compacted tree is correct. A novel linear query processing algorithm which is able to query on the compacted tree without decompression is also introduced. Experiments on both real and synthetic datasets demonstrate that the compression algorithm could compress the original graph by 88% in average, and for denser graphs, the compression algorithm achieves better compression ratio, reducing the original graph by nearly 90%. Comparing with the query processing on original graphs, the query performance on compressed graphs is better, and in average, it could be 1 to 2 orders of magnitude times better.

Key words: maximum Steiner connected *k*-core; 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增大,Facebook 在 2014 年公布其月活跃用户人数(顶点数)达到 12.6 亿,好友关系(边数)达到 1 400 亿,用户数据信息超过 300PB.在这样规模的图上做查询,将是非常耗时的[1].

在现实应用中,社团查询是一种非常普遍的操作.比如:在社交网络中,查询联系紧密的社团可以帮助挖掘

* 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(HIT.NSRIF.201180)

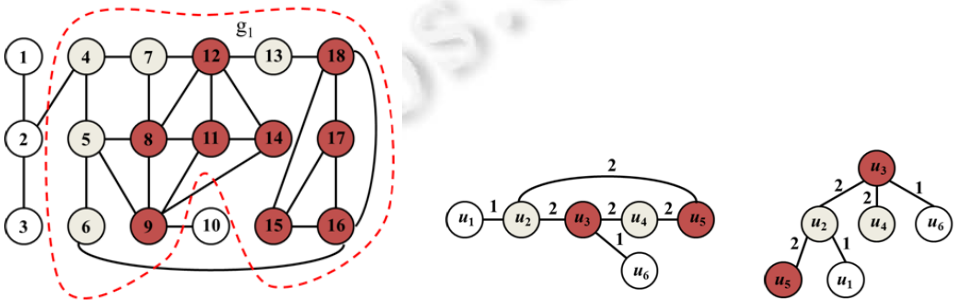
Foundation item: National Basic Research Program of China (973) (2012CB316200); National Natural Science Foundation of China (61190115, 61033015, 61173023); Fundamental Research Funds for the Central Universities (HIT.NSRIF.201180)

收稿时间: 2015-09-24; 修改时间: 2016-01-12; 采用时间: 2016-02-22

用户的社会行为[2];在作者合作网络中,一群联系密切的人可能具有相近的研究领域[3];在购物数据中,一组关系密切的商品很可能被兴趣相近的客户所关注[4];在蛋白质网络中,一些关系紧密的蛋白质往往可能共同形成特定的官能团[5].

目前,对于紧密社团[6]的定义有许多种,如:团、k 团、γ准团、k 丛等.上述问题均是 NP-hard 的,故计算以上的紧密社团将非常耗时.而 k 核作为一种社团的定义,则可以被有效地在线性时间内计算.对于无向图 $G=(V,E)$,k 核是图 G 的一个极大子图,满足子图内的顶点都至少有 k 个邻居出现在子图中.例如在图 1(a)中,{8,9,11,12,14} 的导出子图就是一个 3 核,因为每个顶点在导出子图中都至少有 3 个邻居.作为一种紧密社团,k 核有着广泛的应用:k 核可以作为一种拓扑结构信息用于网络分析[7]和网络可视化[8];k–1 核还可以用于近似表示 k 团;k 核还可以用于给出最稠密子图的 1/2 近似[9]以及最稠密至少 k 子图的 1/3 近似算法[10].

事实上,在很多应用中,用户往往更加关心包含一组给定查询顶点 Q 的连通的 k 核.由于满足上述要求的 k 核并不唯一,而其中最稠密的往往最有意义,故本文所关注的是包含给定查询顶点集 Q 且具有最大 k 值的连通 k 核,简称为 k-MSC.仍以图 1(a)为例,若给定查询顶点集 Q={4,8,15},那么包含 Q 的 k-MSC 为 g_1,是一个 2 核.



(a) 原始图 G (b) 压缩图 G_C (c) 压缩树 G_T

Fig.1 Instances of graph compression

图 1 图压缩实例

本文首先提出了 k-MSC 问题,而现有方法并不能有效地解决 k-MSC 问题.这是因为:(1) 枚举包含 Q 的所有导出子图,并找到其中 k 值最大的连通 k 核,将访问指数级别数量的子图,故并不可行;(2) 借助于现有计算 k 核的最快算法[11],一种直观的算法(Base)可以通过不断递减 k 的值,然后检查是否存在一个连通的 k 核包含了 Q 中的全部顶点.为保证连通性,上述过程每次都需要遍历整个图,当原始图规模很大时,该计算将变得非常耗时.

本文使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-MSC 问题.其思想是:首先将图 G 进行压缩,得到一个规模很小的压缩图 G_C;然后,对于任意 G 上的 k-MSC 查询(G,Q),将其转换成 G_C 上的等价查询(G_C,Q_C),使得 G 上对 Q 的查询结果与 G_C 上对 Q_C 的查询结果完全相同.由于 G_C 的规模远小于 G,故可以大幅提高查询处理效率,并节省存储空间.

文献[12–14]中的图压缩技术将图中的邻接信息用 bit 表示,并通过适当的合并、调整顺序等操作令所使用的 bit 数最少,从而使得其对网络图和社交网络的压缩效果非常可观.然而上述方法仅仅关注了存储代价的降低而忽视了查询效率,这导致压缩图并不能直接被使用,即使是最简单的邻居查询,也需要将压缩图进行适当的解压缩才能够进行查询.除了上述通用的图压缩技术,一些针对特定查询的图压缩技术也被提出[15−18].文献[15]提出了针对可达查询和图模拟查询的图压缩技术;文献[16]探讨了在 XML 树中能够处理 XPath 查询的压缩方法;文献[17]针对邻居查询,提出了具有误差保证的有损压缩技术.然而,由于查询结构的差异,上述方法就不能适用于 k-MSC 查询.

本文首先提出了图压缩算法 SC,其时间复杂度为 $O(|E|)$.SC 算法是基于等价类划分的,故查询转换的时间复杂度为 $O(|Q|)$.本文证明了基于压缩图 G_C 的查询正确性.通过实验验证,SC 算法可以有效地将原始图压缩掉一半以上;且对于稠密图,压缩效果将进一步提升,达到压缩掉近 70%.对于道路交通网络和基于偏好模型的虚拟

图,SC 算法可将原始图压缩掉近 90%.

由于 k-MSC 查询仅需要找到符合要求的连通区域,故 SC 算法得到的压缩图 G_C 还可被进一步压缩.本文提出了图压缩算法 TC,将压缩图 G_C 进一步压缩为树 G_T,其时间复杂度为 $O(|V_C|+|E_C|)$.本文证明了基于压缩图 G_T 的查询的正确性,并提出了压缩树上无需解压缩的查询算法 QueryGT,其时间复杂度为 $O(|R_Q|)$.其中,$|R_Q|$为查询结果集的大小.通过实验验证:在真实数据集上,TC 算法可进一步将 SC 算法的压缩比提升 3~5 倍.与直接在原始图上查询的 Base 算法相比,基于压缩树的查询算法 QueryGT 的查询效率将提升 1~2 个数量级.

本文的主要工作及贡献如下.

(1) 提出了一种支持 k-MSC 查询的图压缩算法 SC 以及相应的查询转换算法,证明了其正确性,SC 算法的时间复杂度为 $O(|E|)$;

(2) 在 SC 算法的基础上,提出了图压缩算法 TC,将压缩图进一步压缩为树,证明了压缩树上查询结果的正确性,TC 算法的时间复杂度为 $O(|V_C|+|E_C|)$;

(3) 提出了压缩树上的查询算法,其时间复杂度为 $O(|R_Q|)$;

(4) 分别在真实数据和虚拟数据上验证了所提出图压缩算法的有效性.真实数据集上的实验结果表明:压缩算法的压缩比可达到 12%;而对于稠密图,算法将取得更好的接近 10%的压缩比.算法对于虚拟数据集同样有效,平均压缩比仍然在 12%左右;且压缩效果随着图稠密度的增加而提高.对平均度数为 30 的虚拟图,压缩比为 6.7%.基于压缩算法的查询算法效率也得到了很好的提升,无论在真实还是虚拟数据集上,查询效率与 Base 算法相比均提高了 1~2 个数量级.

1 相关工作

网络图和社交网络的通用图压缩技术已经被广泛研究,文献[12]的压缩技术能够有效地将网络图压缩,文献[13,14]也可以将社交网络以 bit 的形式有效存储.由于上述技术保存了原始图中全部的结构信息,即,并不针对特定查询,且查询前必须进行解压缩操作,故反而降低了查询效率.

面对特定查询且无需解压缩即可查询的图压缩技术也被提出:文献[15]提出了能够处理可达查询和图模拟查询的图压缩技术,文献[16]利用了双向模拟的思想提出了针对 XPath 查询的压缩技术,文献[17]则针对邻居查询给出了一种具有误差保证的有损压缩技术.然而上述技术仅能处理特定的查询,因此无法适用于 k-MSC 查询.

对于给定图和一组查询顶点 Q,围绕如何找到图中包含 Q 的社团也展开了一些研究[18−21].其中,文献[18]提出了一种基于局部模块化的社团计算方法,文献[19]定义了基于 k-truss 的社团并给出了求解算法,文献[20]研究了基于γ准团定义的社团发现问题.由于对社团定义的差别,上述技术无法适用于 k-MSC 查询.文献[21]研究了基于连通度定义的最大连通社团发现问题,并提出了 SMCC 索引提高查询性能.本文将以 SMCC 索引作为对比,说明 SC 和 TC 算法的有效性.

2 最大 Steiner 连通 k 核

本文针对简单的无向无权图进行研究,即对于给定图 $G=(V,E)$,V 是顶点集,E 是边集,那么图中不存在自环 (u,u),其中,$u\in V$.本文使用的图数据是以邻接表形式存储的,因此,图的规模$|G|$为邻接表中邻接顶点对的数量.对于顶点 $u\in V,v\in V$,(u,v)和(v,u)表示同一条边,而由于(u,v)和(v,u)分别在顶点 u 和 v 的邻接表中各出现一次,故图的规模$|G|=2|E|$.

对于图 G 中的任意的顶点集 H,$G[H]$表示 H 的导出子图,即 $G[H]=(H,\{(u,v)\in E|u,v\in H\})$.记图 G 中连接顶点 u 和 v 的简单路径为 $P(G,u,v)$,则 $P(G,u,v)=\{u,w_1,w_2,\ldots,w_t,v\}$,其中,$(u,w_1)\in E,(w_t,v)\in E,(w_i,w_{i+1})\in E,0<i<t$,当 G 明确时,$P(G,u,v)$简记为 $P(u,v)$.

定义 1. 给定图 $G=(V,E)$,k 核是满足以下性质的子图 $G[H]$:(1) $H\subseteq V$;(2) 对于顶点 $u\in H$,u 在 $G[H]$中至少有 k 个邻居;(3) $G[H]$是极大的,即,不存在 H',满足 $H\subseteq H'\subseteq V$ 且 $G[H']$是 k 核.

定义 2. 给定图 $G=(V,E)$,对于顶点 $u\in V$,u 的核值 $core(u)$为 u 所在全部 k 核中的最大值,即:

$core(u)=\mathrm{Max}_{H\subseteq V}\{k|u\in H$ 且 $G[H]$为 k 核$\}$.

例如,在图 1(a)中,由 $H=\{8,9,11,12,14\}$生成的导出子图中每个顶点的核值均为 3,因为它们在 $G[H]$中的度均为 3 且图中不存在一个 4 核.但是 $G[H]$并不是 3 核,因为图中存在着一个更大的 3 核 $G[H']$,其中 $H'=\{8,9,11,12,14,15,16,17,18\}$.值得注意的是,$k$ 核并不一定是连通的.

定义 3. 给定图 $G=(V,E)$,查询顶点集 Q,包含 Q 的 Steiner 连通 k 核是满足以下性质的子图 $G[H]$:(1) $Q\subseteq H\subseteq V$ 且 $G[H]$是连通的;(2) 对于顶点 $u\in H$,u 在 $G[H]$中至少有 k 个邻居;(3) $G[H]$是极大的,即,不存在 H',满足 $H\subseteq H'\subseteq V$ 且 $G[H']$是 Steiner 连通 k 核.本文简称 Steiner 连通 k 核为 k-SC.

定义 4. 给定图 $G=(V,E)$,查询顶点集 Q,包含 Q 的最大 Steiner 连通 k 核是核值最大的 k-SC,简称为 k-MSC,记此最大核值为 MSC(G,Q).当 G 明确时,MSC(G,Q)可简记为 MSC(Q).

仍然以图 1(a)为例,若 $Q=\{8\}$,则存在一个 3-SC,$G[\{8,9,11,12,14\}]$,一个 2-SC,$G[V\setminus\{1,2,3,10\}]$和一个 1-SC,$G[V]$.根据定义 4 可知:包含{8}的 k-MSC 为 $G[\{8,9,11,12,14\}]$,而 MSC$(\{8\})=3$.

事实上,当$|Q|=1$,即,查询顶点集只包含一个顶点时,记 Q 中顶点为 u,则 MSC$(Q)=core(u)$,包含 Q 的 k-MSC 为 $core(u)$核中包含 u 的极大连通子图.仍以 $Q=\{8\}$为例,显然,MSC$(\{8\})=core(8)=3$ 成立.而由于图中的 3 核为 $G[\{8,9,11,12,14,15,16,17,18\}]$,故包含{8}的极大连通子图为 $G[\{8,9,11,12,14\}]$,$G[\{8,9,11,12,14\}]$也是包含{8}的 k-MSC.

由于 k-SC 和 k-MSC 均为导出子图,故在不引起歧义时,可以将 k-SC 和 k-MSC 看作顶点集 H.在下文中,将交替使用 H 和 $G[H]$表示 k-SC 和 k-MSC 的查询结果.另外,由于计算包含 Q 的 k-MSC 和 MSC(Q 的过程非常相近,且计算前者需要额外的步骤,故本文仅讨论前种查询.

下面给出 G 的顶点集 V 上等价关系"≡"的定义.

定义 5. 给定图 $G=(V,E)$,"≡"是 V 上的二元关系,对于顶点 $u\in V,v\in V,u\equiv v$ 当且仅当图 G 中存在一条简单路径 $P(u,v)$,使得对于任意顶点 $w\in P(u,v)$,$core(w)=core(u)=core(v)$.

由于任意顶点均与自身连通,故 $u\equiv u$.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,$[u]_C=\{v|v\in V,u\equiv v\}$表示包含顶点 u 的等价类,若顶点 v 与等价类 $[u]_C$ 中任意顶点 u 等价,则称 v 属于$[u]_C$,记作 $v\sqsubseteq[u]_C$.

2.1 图压缩算法SC

本文首先给出第 1 种图压缩方法 SC: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 $G_C=(V_C,E_C,f)$中的一个顶点;压缩图 G_C 中两个顶点$[u]_C$和$[v]_C$之间存在一条边$([u]_C,[v]_C)$当且仅当在原始图 G 中存在两个顶点 $u\in[u]_C$ 和 $v\in[v]_C$,满足$(u,v)\in E$,f 是顶点集上 V_C 的权值函数,记录了该顶点对应的原始图顶点的核值.算法 1 给出了 SC 算法的执行过程.关于算法 1 的具体细节和复杂度分析,将在第 2.2 节介绍.

在将图 G 压缩为 G_C 后,可以在压缩图 G_C 上直接查询.对于任意给定查询顶点集 Q,$Q_C=\{v|u\sqsubseteq v,u\in Q\}$是压缩图上与 Q 对应的查询顶点集,即,Q 中顶点所在等价类的集合.为了计算图 G 中包含 Q 的 k-MSC,可以通过计算 G_C 中包含 Q_C 的 k-MSC 作为查询结果.

例如在对图 1(a)中的原始图 G 进行压缩时,因为顶点 1,2,3 核值均为 1 且可以通过路径 $P(1,3)=\{1,2,3\}$连接,所以合并 1,2,3 得到 u_1 且 $f(u_1)=1$,如图 1(b)所示($u_1=\{1,2,3\}$,$u_2=\{4,5,6,7\}$,$u_3=\{8,9,11,12,14\}$,$u_4=\{13\}$,$u_5=\{15,16,17,18\}$,$u_6=\{10\}$),类似地,顶点 4,5,6,7 可合并为 u_2 且 $f(u_2)=2$,8,9,11,12,14 可合并为 u_3 且 $f(u_3)=3$,15,6,17,18 可合并为 u_5 且 $f(u_5)=3$,至此可得到图 1(b)中的压缩图 G_C.而对于给定查询"包含{8,12,15}的 k-MSC",只需计算压缩图 G_C 中包含$\{u_3,u_5\}$的 k-MSC 即可,而包含$\{u_3,u_5\}$的 k-MSC 为 $G_C[\{u_2,u_3,u_4,u_5\}]$,这与原始图 G 上的查询结果是相同的.

以下定理将证明在压缩图 G_C 上计算的 k-MSC 的是正确的.

引理 1. 在图 $G=(V,E)$中,若$(u,v)\in E$ 且 $core(u)=core(v)$,则对于图 G 中任意一个最大 Steiner 连通 k 核 $G[H]$,若 $u\in H$,则 $v\in H$.

证明:记 $G[H]$的核值为 p,若 $u\in H$,则 $p\leqslant core(u)$.不妨假设包含$\{v\}$的 k-MSC 为 $G[H']$,若 $H\neq H'$,那么考查由

$H\cup H'$生成的导出子图 $G[H\cup H']$:

(1) $G[H\cup H']$一定是连通的.这是由于 $G[H]$和 $G[H']$均是连通的并且二者通过(u,v)连接;

(2) 对于顶点 $w\in H\cup H'$,w 在 $G[H\cup H']$中一定至少有 p 个邻居.这是由于 $G[H]$中顶点的度至少为 p 且 $G[H']$中顶点的度至少为 $core(v)\geqslant p$,而 $G[H\cup H']$中顶点的度不会小于前两者.

因此,我们找到了一个 $G[H]$的超图 $G[H\cup H']$并且 $G[H\cup H']$是 p-SC,矛盾!故而 $H=H'$,即,顶点 u 和 v 一定出现在同一个 k-MSC 中. □

结论 1. 在图 $G=(V,E)$中,若 $u\equiv v$,则对于图 G 中任意一个最大 Steiner 连通 k 核 $G[H]$,若 $u\in H$,则 $v\in H$.

证明:根据引理 1,易证该结论. □

根据上述结论可知,属于相同等价类的顶点一定会出现在相同的 k-MSC 中.这说明本文中等价关系的定义是合理的.

定理 1. 给定图 $G=(V,E)$,$G_C=(V_C,E_C,f)$是由 SC 算法计算的压缩图,若 H_C 是 G_C 中的一个 k-MSC,则 H 是 G 中的一个 k-MSC,其中,$H=\{v|v\sqsubseteq u,u\in H_C\}$.

证明:用反证法.若 $G[H]$不是一个 k-MSC,则图 G 中要么存在一个 $G[H]$的超图 $G[H']$,使得 $G[H']$是 k-SC;要么存在一个 $G[H'']$,使得 $G[H'']$的核值比 $G[H]$更大.

首先讨论第 1 种情况.由于 $G[H']$是连通的,不妨假设 $w\in H'\backslash H$ 且 w 与 H 相邻.显然,w 对应的等价类$[w]_C\notin H_C$.根据结论 1 可以发现:(1) $G_C[H_C\cup\{[w]_C\}]$是连通的;(2) $G_C[H_C\cup\{[w]_C\}]$的核值不小于 $G_C[H_C]$.矛盾!故第一种情况是不存在的.

下面讨论第 2 种情况.若存在一个核值更大的 $G[H'']$,则记 $H'_C=\{u\in V_C|v\sqsubseteq u,v\in H''\}$.根据 SC 算法,若$(u,v)\in E$,则$([u]_C,[v]_C)\in E_C$,故 $G_C[H'_C]$一定是连通的.因此,我们在压缩图 G_C 中找到了一个核值更大的 k-SC,这意味着 G_C 中一定存在着一个核值更大的 k-MSC.矛盾!故第 2 种情况也是不存在的.

综上所述,可证结论. □

定理 2. 给定有向图 $G=(V,E)$,查询顶点集 Q,$G_C=(V_C,E_C,f)$是由 SC 算法计算的压缩图,Q_C 是 Q 对应的等价类集,则在原始图 G 上包含 Q 的 k-MSC 与在压缩图 G_C 上包含 Q_C 的 k-MSC 是相同的.

证明:根据定理 1,在压缩图 G_C 上包含 Q_C 的 k-MSC 一定是原始图 G 中包含$\{v|v\sqsubseteq u,u\in Q_C\}$的 k-MSC.下面只需证明在原始图中,查询顶点集 Q 和$\{v|v\sqsubseteq u,u\in Q_C\}$具有着相同的 k-MSC.

根据结论 1,同一等价类中的顶点一定在同一个 k-MSC 中,故定理 2 得证. □

2.2 SC算法的分析

SC 算法采用了基于等价类的压缩思想,因此,如何有效地划分等价类是 SC 算法的关键.

首先需要计算图中所有顶点的核值,根据文献[11]中的算法,通过使用桶排序,所有顶点核值的计算可以在 $O(|E|)$内完成.见算法 1 第 1 行.

其次,为了划分等价类,将依次为每一个顶点分配唯一的类别.根据定义 5,$u\equiv v$ 当且仅当图中存在 $P(u,v)$,使得对于任意顶点 $w\in P(u,v)$,$core(w)=core(u)=core(v)$.因此可以从任意的顶点 u 出发,采用广度优先搜索算法,不断地扩展与顶点 u 核值相同的顶点,直至无法扩展为止.将上述顶点集划分为同一等价类$[u]_C$,同时标记为已访问.至此,顶点 u 所在的等价类划分完毕.对于图中剩余的等价类,只需要不断重复上述过程,直至图中所有顶点均被访问过为止,见算法 1 第 3 行~第 11 行.显然,基于 BFS 的等价类划分可以在 $O(|E|)$的时间内完成.

最后,通过依次访问原始图 G 中的每一条边,可以确定压缩图的边集.显然,生成压缩图仍然可以在 $O(|E|)$内完成.故,图压缩算法 SC 的时间复杂度为 $O(|E|)$.

算法 1. SC.

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

Output: Compressed Graph $G_C=(V_C,E_C,f)$, Map List M.

① *Kcore*();

② $V_C=E_C=\varnothing$;
③ **For** each $u\in V$ **do**
④ **If** u is not visited **then**
⑤ BFS($core(u)$); //仅扩展与 u 核值相同的顶点
⑥ **For** each v expanded by BFS($core(u)$) **do**
⑦ $M[v]=[u]_C$;
⑧ v is visited;
⑨ **end for**
⑩ $V_C=V_C\cup[u]_C$, $f([u]_C)=core(u)$;
⑪ $M[u]=[u]_C$, u is visited;
⑫ **end if**
⑬ **end for**
⑭ **For** each vertex pair $[u]_C\in V_C$, $[v]_C\in V_C$ **do**
//计算压缩图的边集
⑮ **if** $(u,v)\in E$ **then**
⑯ $E_C=E_C\cup([u]_C,[v]_C)$;
⑰ **end if**
⑱ **end for**
⑲ **return** $G_C=(V_C,E_C)$, M;

值得注意的是:图压缩算法 SC 在返回压缩图 G_C 的同时还将返回一个映射表 M,M 记录了原始图中每一个顶点所对应的等价类.借助于 M,对于给定查询(G,Q),可以在 $O(|Q|)$时间内将之转换为(G_C,Q_C),以用于进一步的查询处理.由于 M 的规模为 $O(|V|)$,故可将 M 存放于内存,且由于 M 的大小并不影响查询速度,故不将 M 计入压缩图.

另外,压缩图需要额外保存顶点的核值信息.这是由于压缩后顶点的邻居信息不再完整,即:如果不保存核值信息,那么将无法从压缩图得到正确的 k-MSC 结果.而在真实的图数据中,顶点的核值往往是比较小的,比如在 Brightkite 和 Gowalla 两个社交网络中,顶点的最大核值分别为 52 和 51.而在道路交通网络中,顶点的核值更小,比如 roadNet-CA 网络中,顶点最大核值仅为 3.这说明压缩图中用于保存顶点权值所占的空间是很小的,故在考查压缩比时,仍然以$|E_C|/|E|$为准.

3 最大生成树压缩算法 TC

本文将在 SC 算法的基础上给出第 2 种图压缩算法 TC,从而将压缩图 G_C 进一步压缩.TC 算法的大致思想为:将压缩图 G_C 中的边进行适当选择,得到一棵树 $G_T=(V_T,E_T,f)$,使得 $V_T=V_C$;且对于查询(G_C,Q_C),通过计算(G_T,Q_C)可以得到相同的查询结果.

上述思想来自于对压缩图 G_C 上查询过程的观察.在计算包含 Q_C 的 k-MSC 时,可以分为两个步骤:(1) 找到包含 Q_C 的一个连通区域 C,记 C 的权值为 C 中顶点权值最小者,即 $f(C)=\mathrm{Min}\{f(u)|u\in C\}$,那么查询所要找到的 C 是所有包含 Q_C 的连通区域中权值最大者;(2) 在保持权值不变和连通的前提下,将 C 扩展至极大,得到 H,H 即为包含 Q_C 的 k-MSC.根据 k-MSC 的定义和结论 1,上述查询结果显然是正确的.由于步骤(2)依然可以使用广度优先搜索算法实现,故问题的关键在于如何发现包含 Q_C 且权值最大的连通区域 C.

计算 C 可以进一步地分解为不断地构建连接 Q_C 中顶点对的简单路径.记 $Q_C=\{u_1,u_2,\ldots,u_p\}$,那么计算 C 就可以看作依次构建 p–1 条简单路径:$P(u_1,u_2)$,$P(u_1,u_3)$,…,$P(u_1,u_p)$.记每条简单路径 P 的权值为 P 上顶点权值最小者,即 $f(P)=\mathrm{Min}\{f(u)|u\in P\}$,那么查询所要找到的每一条 $P(u_1,u_i)$都是连接顶点 u_1 和 u_i 的所有简单路径中权值最大者.

以图 1(b)中的压缩图 G_C 为例,若查询顶点集为{8,12,15},则转换后可知 $Q_C=\{u_3,u_5\}$.那么查询过程分为以下两步:首先,找到包含$\{u_3,u_5\}$的权值最大的连通区域,该过程可以进一步转换为计算权值最大的 $P(u_3,u_5)$,易知,路径(u_3,u_4,u_5)是符合条件的,因此 $C=\{u_3,u_4,u_5\}$且$f(C)=2$;最后,在保持连通性和权值不变的前提下,将 C 进一步扩展为$\{u_2,u_3,u_4,u_5\}$,那么 $G_C[\{u_2,u_3,u_4,u_5\}]$就是最终的查询结果.

定义 6. 给定图 $G=(V,E,W)$,W 是边集 E 上的权值函数,G 的子图 G_1 是一棵生成树当且仅当 G_1 是树且 G_1 包含了 G 中全部顶点,若 G_1 在所有生成树中具有最大的权值和,则 G_1 为最大生成树.

下面证明 $G_C=(V_C,E_C,f)$的最大生成树 $G_T=(V_T,E_T,f)$满足本文的查询要求,即:对于查询(G_C,Q_C),计算(G_T,Q_C)可以得到相同的查询结果.这里需要强调的一点是:对于 E_C 中每条边(u,v)的权值,取(u,v)两端的顶点中权值较小者.这是因为当路径 P 经过(u,v)时,$f(P)$一定不会超过 u 和 v 中权值较小者.

定理 3. 给定图 $G_C=(V_C,E_C,f)$,$G_T=(V_T,E_T,f)$是 G_C 的最大生成树,则对于顶点 $u\in V_C,v\in V_C,P(G_T,u,v)$是图 G_C 中连接 u 和 v 的权值最大的简单路径.

证明:用反证法.假设 $P(G_T,u,v)$在图 G_C 中不是权值最大的简单路径,那么一定存在一条权值更大的路径 P'.下面通过例子来说明.以图 1(c)为例,不妨设 $P(G_T,u,v)=(u_2,u_3,u_6)$,$P'=(u_2,u_4,u_6)$.那么向 G_T 中增加两条边(u_2,u_4)和(u_4,u_6),此时,G_T 中至少产生了两个环.首先找到路径 $P(G_T,u,v)$上权值最小的边(u_3,u_6)并将之删除;然后,G_T 中还存在着一个环.继续删除剩余环中权值最小的边(u_2,u_3),得到一棵新的生成树 G_T'.

在上述过程中,可以将插入边和删除边的操作配对,即:首先插入边(u_4,u_6),随后产生环(u_3,u_4,u_6),再删除边(u_3,u_6)破坏环得到新的生成树;然后插入边(u_2,u_4),产生环(u_2,u_3,u_4),再删除边(u_2,u_3)得到 G_T'.

因为 P'的权值大于 $P(G_T,u,v)$,故首次产生环并破坏环的操作一定使生成树的权值和增大.又因为每次产生环并破坏环的操作一定不会令生成树的权值和减小,故 G_T' 的权值和一定大于 G_T,所以 G_T 不是最大生成树.矛盾!故结论成立. □

定理 4. 给定图 $G_C=(V_C,E_C,f)$,$G_T=(V_T,E_T,f)$是 G_C 的最大生成树,则(G_C,Q_C)与(G_T,Q_C)的查询结果相同.

证明:用构造法证明.根据定理 3,对于 $u\in Q_C,v\in Q_C,P(G_T,u,v)$一定是图 G_C 中连接 u 和 v 的权值最大的简单路径,那么对于 $Q_C=\{u_1,u_2,\ldots,u_p\}$,通过依次构建 $p-1$ 条简单路径:$P(G_T,u_1,u_2)$,$P(G_T,u_1,u_3)$,…,$P(G_T,u_1,u_p)$,并据此形成的连通区域 $C=\{v|v\in P(G_T,u_1,u_i),0<i<p\}$一定是所有包含 Q_C 的连通区域中权值最大的.然后,在保持连通性和权值不变的前提下,将 C 扩展至极大至 H,显然,$G_C[H]$即为包含 Q_C 的 k-MSC.

故结论成立. □

下面分析图压缩算法 TC 的时间复杂度.根据文献[22],在使用斐波那契堆进行排序时,最大生成树算法可以在 $O(|E|+|V|\log|V|)$内完成.注意到压缩图 G_C 边上的权值均为顶点的核值,即整数,故本文将采用桶排序算法,从而将 TC 算法的时间复杂度降至 $O(|V_C|+|E_C|)$.

下面介绍本文所使用桶的数据结构.桶以链表的形式存储,每个桶对应着一个权值,即顶点核值.为了提高插入删除操作的效率,还需要两个额外的数据结构:首先,为了提高插入效率,为每一个桶记录一个尾部指针,当桶中有新的元素加入时,只需在尾部增加即可,故插入的时间复杂度为 $O(1)$;其次,为了提高删除效率,为每一个桶内的元素记录其父亲元素的指针,当该元素要从桶中删除时,只需将令父亲指向其儿子即可,故删除的时间复杂度为 $O(1)$.由于 TC 算法执行过程中至多向桶中插入/删除$|E_C|$次,故可将 TC 算法的时间复杂度降至 $O(|V_C|+|E_C|)$.算法 2 中给出了 TC 算法的具体执行过程.

算法 2. TC.

Input: Compressed Graph $G_C=(V_C,E_C,f)$;

Output: Compressed Tree $G_T=(V_T,E_T,f)$.

① **For** each $u\in V$ **do**

② **For** each $v\in N(u)$ **do**

③ insert (u,v) into Bins;

④ **end for**

⑤ u is visited;
⑥ **while** Bins is not empty **do**
⑦ extract (x,y) with maximum value, ;
//此处认为 x 已被访问过,y 未被访问过
⑧ insert (x,y) into E_T;
⑨ **For** each $v \in N(y)$ **do**
⑩ **if** v is not visited **then**
⑪ insert (y,v) into Bins;
⑫ **else**
⑬ delete (y,v) from Bins;
⑭ **end if**
⑮ **end for**
⑯ y is visited;
⑰ **end while**
⑱ **end for**
⑲ **Return** $G_T=(V_C,E_T,f)$;

最后介绍关于在压缩图 $G_T=(V_T,E_T,f)$上的查询算法.因为压缩图 G_T 是一棵树,故可以有效地利用树的结构提高查询效率.在查询前,可以首先作以下预处理:

(1) 选取度值最大的顶点作为根节点,当出现度值相等的情况时,选取核值大的顶点作为根节点.这样做的好处是可以让树尽可能均衡,因为度值大意味着树的深度较小,而核值大意味着该顶点可出现在更多的 k-SC 中;

(2) 计算每个顶点到根的距离,即顶点的深度.显然,预处理的时间复杂度为 $O(|V|+|E|)$,而上述预处理的结果均不会存储于压缩图中.

压缩图 G_T 上的查询算法 QueryGT 见算法 3.首先,依次计算 $Q_C=\{u_1,u_2,\ldots,u_p\}$中相邻顶点 u_i 和 u_{i+1} 的最小公共祖先[23]:LCA(u_i,u_{i+1}),其中,LCA(u_i,u_{i+1})是指在一棵有根的树中,u_i 和 u_{i+1} 的公共祖先中距离根节点最远的.在计算出 LCA(u_1,u_2)后,记 u_2=LCA(u_1,u_2)并继续计算 LCA(u_2,u_3),然后记 u_3=LCA(u_2,u_3)并继续计算 LCA(u_3,u_4),以此类推.显然,最终得到的 LCA 一定是顶点集 Q_C 中所有顶点的最小公共祖先,记作 LCA(Q_C).而借助于预处理中顶点的深度信息,上述过程所访问的所有顶点形成了一棵以 LCA(Q_C)为根的子树,且该子树一定是包含 Q_C 的 k-MSC 的子图.事实上,该子树正是前文所提到的包含 Q_C 的权值最大的连通区域 C.

然后,在保持连通性和权值不变的前提下,对上述子树进行 BFS 扩展,可得到极大子图 $G_C[H]$,即为最终查询结果.显然,QueryGT 算法仅需访问查询结果集中的顶点,故其时间复杂度为 $O(|R_Q|)$.

算法 3. QueryGT.

Input: Compressed Graph $G_T=(V_T,E_T,f)$, Query Set Q, Map List M;

Output: k-MSC H which contains Q.

① compute Q_C based on Q and M, min=$|V_T|-1$, $H=\varnothing$; ;
② **for** $0<i<|Q_C|$ **do**
③ compute LCA(u_i,u_{i+1});
④ **if** v is visited during LCA computation **then**
⑤ v is visited, min=Min{$f(v)$,min}, $H=H\cup\{v\}$;
⑥ **end if**
⑦ u_{i+1}=LCA(u_i,u_{i+1});
⑧ **end for**

⑨ *push*(*Queue*,LCA(Q_C)); //扩展计算极大子图
⑩ **while** *Queue* is not empty **do**
⑪ *u*=*pop*(*Queue*);
⑫ **for** each $v \in N(u)$ **do**
⑬ **If** v is not visited and $f(v) \geqslant$min **then**
⑭ *push*(*Queue*,*v*), v is visited, $H=H\cup\{v\}$;
⑮ **end if**
⑯ **end for**
⑰ **end while**
⑱ **Return** H;

4 实验结果

在实验部分,本文在真实数据集上对两种图压缩算法 SC 和 TC 的有效性和查询效率进行了考查.在考查压缩比时,使用 SMCC 索引作为对比;在考查查询效率时,使用了直接在原始图上查询的直观算法 Base 的查询算法作为对比.为验证 SC 和 TC 算法的扩展性,本文还在虚拟数据集上检验了两种算法的压缩比和查询效率.

下面首先介绍实验考查的几种测度,包括压缩算法压缩比和查询处理时间;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.

在考查算法的压缩比时,分别用 CR_{SC} 和 CR_{TC} 表示压缩算法 SC 和 TC 算法的压缩比,其中,$CR_{SC}=|E_C|/|E|$, $CR_{TC}=|E_T|/|E|$.可知:当压缩比越小时,压缩图的规模也越小,说明压缩算法更加有效.在考查查询处理时间时,对比了基于压缩树的查询处理算法 QueryGT 的时间开销与在原始图上直接进行查询的 Base 算法的时间开销.为公平起见,Base 算法并不计算原始图中顶点的核值,即,认为顶点核值也是 Base 算法的输入.

实验中使用的真实数据集来源于斯坦福大学的共享数据集(http://snap.stanford.edu/data/#twitter).本文使用的数据规模在 404K~23.9M 之间,包括了以下网络:(1) Email-Enron,美国安然公司的邮件网络,其中,顶点表示邮箱,边表示某个邮箱向另一邮箱发送过邮件;(2) ca-AstroPh,天体物理学期刊 Arxiv 中的作者合作网络,其中,顶点表示作者,边表示两位作者合作过;(3) Brightkite 和 Gowalla,在线社交网络,其中,顶点表示用户,边表示朋友关系;(4) roadNet-TX 和 roadNet-CA,美国德克萨斯州和加利福尼亚州的道路交通网络,其中,顶点表示道路交汇点,边表示道路;(5) as-Skitter,互联网拓扑图,其中,顶点表示网页,边表示超链接.表 1 中给出了本文所使用真实数据集的特性,其中,平均度数为$|E|/|V|$,反映了图的稠密程度;最大核值为图中核值最大的顶点,反映了图中最稠密社团的稠密度.实验所使用虚拟图是基于偏好模型生成的[24],即:图中顶点更容易和度数大的顶点共同组成一条边,而不容易和度数小的顶点组成边.为了突出少数顶点的度数,图中将初始化一个规模为平均度数的团,为使虚拟图能适用于实验,将人为去掉图中的自环和多重边.

上述实验使用 C++实现,在双核 3.40GHz Intel Core(PM) D CPU、32.0GB 内存、Window7 系统的台式机上运行.

Table 1 Properties for real datasets
表 1 实验中使用真实数据集的基本特性

数据集	顶点数	边数	平均度数	最大核值
Email-Enron	36 692	367 662	10.02	43
ca-AstroPh	18 772	396 160	21.10	56
Brightkite	58 228	428 156	7.35	52
Gowalla	196 591	1 900 654	9.67	51
roadNet-TX	1 379 917	3 843 ,320	2.79	3
roadNet-CA	1 965 206	5 533 214	2.82	3
as-Skitter	1 696 415	22 190 596	13.08	111

表 2 中给出了 SC,TC 算法和 SMCC 索引在真实数据集上的压缩比.实验结果表明:(1) 图压缩算法的压缩

效果非常可观,且对于稠密图将取得更好的压缩比;(2) TC 算法可以在 SC 算法的基础上进一步有效地压缩原始图.首先可以发现:所提出图压缩算法的平均压缩比为 12%,而对于平均度数大于 10 的 Email-Enron,ca-AstroPh 和 as-Skitter,压缩比则接近 10%.其中,对于平均度数为 21.1 的作者合作网络 ca-AstroPh,压缩图规模仅为原始图的 3.9%.除道路交通网络外,最稀疏的 Brightkite 的压缩效果最差,但是也达到了 17.8%.这是由于稠密图中往往会出现多个稠密区域,而这些稠密区域就很可能会被压缩算法归为同一个等价类,故本文的压缩算法对于稠密图更有效.其次,除去道路网络图,可以发现,TC 算法可以将 SC 算法得到的压缩图 G_C 进一步压缩 3~5 倍.比如:对于 as-skitter,SC 算法可以将原始图压缩至 42.1%,而 TC 算法则可进一步将其压缩至 10.5%;而对于 ca-AstroPh,尽管 SC 算法的压缩比已经比较客观,为 19.4%,但是 TC 算法仍然能进一步将压缩效果提升至 3.9%.这说明两种算法都是必要的.

在道路交通网络中,尽管顶点的平均度数很小(往往不超过 3),但是由于道路交通网络中的稠密区域大部分都很稀疏(这一点可以在表 1 中发现,roadNet-TX 和 roadNet-CA 的最大核值均仅为 3),所以道路交通网络更加容易被基于等价类划分的 SC 算法压缩.可以发现,TC 算法在上述两个数据集上无法将 SC 算法得到的压缩图 G_C 进一步压缩.这说明仅使用 SC 算法,就能够将道路交通网络图充分压缩.

与 SMCC 索引相比,本文所使用的图压缩算法具有更好的压缩性能.SMCC 索引的平均压缩比为 33%,这几乎是所提出算法的 3 倍.可以发现:在 Brightkite,Gowalla 和 as-skitter 上,SMCC 索引的规模更小,仅为压缩图的 1.5 倍;而在 roadNet-TX 和 roadNet-CA 上,SMCC 索引的规模都超过了原始图的 70%,这意味着 SMCC 索引几乎无法将原始图压缩.上述情况的出现是由于道路交通网络是十分稀疏且连通的,因此几乎可以看作一棵树,而 SMCC 索引的思想就是将原始图表示成树,故 SMCC 索引几乎不能缩减道路交通网络的规模.

Table 2 Compressed graph size and compression time of SC, TC and index size of SMCC on real datasets

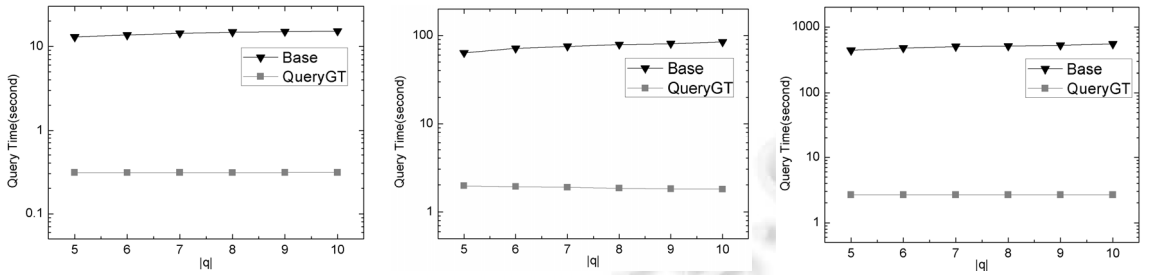
表 2 真实图数据上 SC,TC 的压缩图大小、压缩时间与的 SMCC 索引大小

	数据集	Email-Enron	ca-AstroPh	Brightkite	Gowalla	road-TX	road-CA	as-skitter
SC	顶点数	22 829	7 691	38 228	123 452	224 530	278 301	1 167 322
	边数	125 910	76 962	183 804	811 300	449 034	556 544	9 343 008
	压缩比(%)	34.2	19.4	42.9	42.7	11.7	10.1	42.1
	压缩时间(s)	0.018	0.029	0.032	0.184	0.54	0.80	4.51
TC	顶点数	22 829	7 691	38 228	123 452	224 530	278 301	1 167 322
	边数	45 574	15 334	76 392	246 902	449 034	556 544	2 334 626
	压缩比(%)	12.4	3.9	17.8	13.0	11.7	10.1	10.5
	压缩时间(s)	0.019	0.013	0.025	0.18	0.08	0.11	2.68
SMCC	顶点数	36 692	18 772	58 228	196 591	1 379 917	1 965 206	1 696 415
	边数	71 254	35 826	115 362	393 180	2 758 986	3 925 136	3 391 318
	压缩比(%)	19.4	9.0	26.9	20.7	71.8	70.9	15.3

表 2 还给出了 SC,TC 算法的压缩时间,实验结果表明:两种压缩算法的压缩时间都很快速.对于规模最大的 road-CA 和 as-skitter,压缩算法分别可在 1s 和 8s 内完成.

图 2 给出了 Base 算法与基于压缩图的 QueryGT 算法在真实图上的查询时间对比,所使用的图包括 Gowalla,roadNetTX 和 as-skitter.这是由于邮件网络 Emai-Enrom、作者合作网络 ca-AstroPh 和在线社交网络 Brightkite、Gowalla 均具有相似的社交网络的特性,road-TX 和 road-CA 均为道路网络,而 as-skitter 则代表着网络图,故本文仅以 3 类图数据中规模较大的 Gowalla,road-TX 和 as-skitter 为例考查基于压缩图的查询效率.查询顶点集是 1 000 组随机生成的、大小从 5 增加至 10 的顶点集,查询时间是 1 000 组查询时间的总和.实验结果表明:(1) QueryGT 的查询效率比 Base 提高了一至两个数量级;(2) 两种算法的查询时间均随着给定查询顶点集大小$|Q|$的增大而略有增加.首先可以发现:对于 Gowalla 和 roadNet-TX,QueryGT 的查询速度是 Base 的 50~60 倍;而对于 as-skitter,QueryGT 的查询效率则是 Base 的 200 倍左右.这是由于图压缩算法不仅将原始图进行了有效地压缩,且保证了 QueryGT 在压缩图上查询时仅仅需要访问查询结果集中的顶点,因此查询效率得到极大的提升.其次,当查询顶点集大小$|Q|$从 5 增加至 10 时,两种算法的查询时间均有微小的增加.这是因为随着查询顶

点集的增大,查询结果将有可能变成核值更小、覆盖顶点更多的连通区域,这将使查询结果集变得更大,而两种算法的运行时间均与查询结果集大小相关,故查询时间会随之增加.

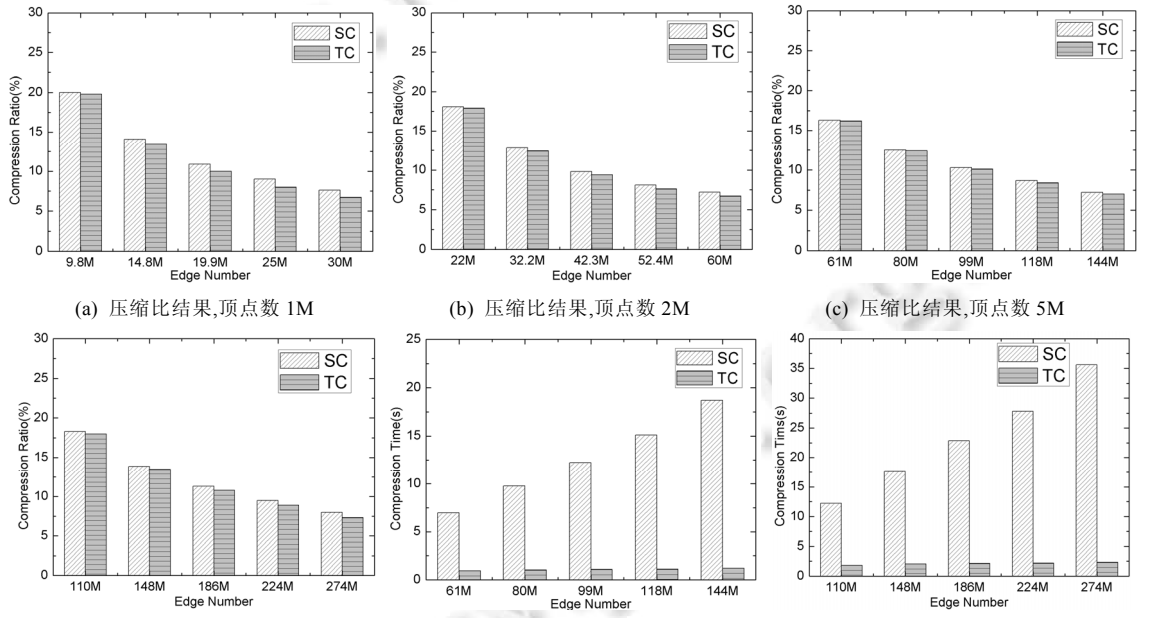


(a) Gowalla 数据集 (b) roadNet-TX 数据集 (c) as-skitter 数据集

Fig.2 Query time of Base vs. QueryGT on real datasets

图 2 真实图数据上查询算法 Base 和 QueryGT 的对比

图 3(a)~图 3(d)给出了 SC 和 TC 算法在虚拟图上的压缩比,为了验证算法的扩展性,分别在顶点数为 1M~10M 时,将虚拟图的边数从 9.8M 增加至 274M.



(a) 压缩比结果,顶点数 1M (b) 压缩比结果,顶点数 2M (c) 压缩比结果,顶点数 5M

(d) 压缩比结果,顶点数 10M (e) 压缩时间,顶点数 5M (f) 压缩时间,顶点数 10M

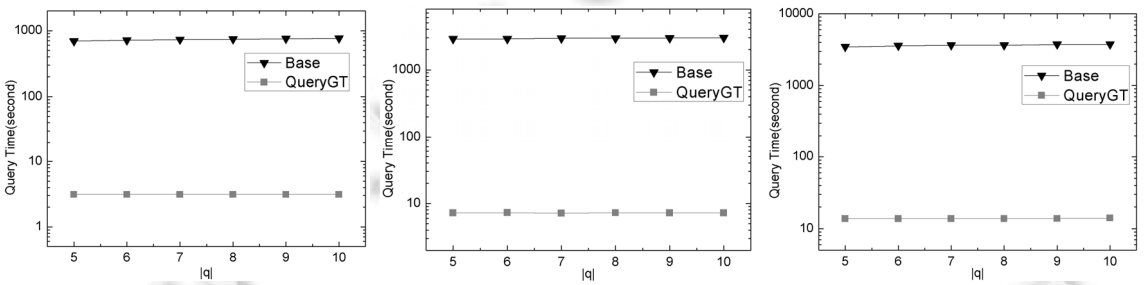
Fig.3 Compression ratio and compression time of SC, TC on synthetic datasets

图 3 虚拟图数据上 SC, TC 算法的压缩比与压缩时间

实验结果表明:图压缩算法在虚拟图上仍然就有很好的压缩效果;且随着平均度数的增加,压缩图的规模更小.即,算法更适用于稠密图.首先可以发现:压缩算法在顶点数为 1M~10M 的虚拟图上都取得了平均低于 12%的压缩比,这说明压缩算法可以将虚拟图有效地压缩;当虚拟图的平均度数为 20 左右时(即边数为 19.9M, 42.3M,99.3M,224M 时),两张虚拟图压缩后的规模均在原始图规模的 10%左右.其次,当顶点数固定、边数增加时,算法的压缩比从 19.8%降至 6.7%,且保持了单调下降的趋势.这说明压缩算法更加适用于稠密图.值得注意的一点是:在虚拟图上,SC 和 TC 算法的压缩效果极为接近,这是由于在偏好模型下,少数度数较大的顶点将构成一

个等价类,而剩余小度数的顶点经过 SC 算法的压缩将非常稀疏,即接近于树,因此 TC 算法无法将上述压缩图进一步大幅度压缩.这说明 SC 算法在本文所使用的虚拟图模型下更加适用.图 3(e)和图 3(f)给出了 SC,TC 算法在虚拟图上的压缩时间,实验结果表明,两种算法的压缩时间仍然都很快速.对于顶点数为 5M 和 10M、边数为 61M~274M 的虚拟图,压缩算法可在 40s 内完成.

图 4 给出了 Base 算法与基于压缩图的 QueryGT 算法在虚拟图上的查询时间对比,其中,查询顶点集是 1 000 组随机生成的、大小从 5 增加至 10 的顶点集,查询时间是 1 000 组查询时间的总和.实验结果表明:(1) QueryGT 的查询效率比 Base 提高了两个数量级;(2) 两种算法的查询时间均随着给定查询顶点集大小$|Q|$的增大而略有增加.首先可以发现:当顶点数分别为 2M,5M 和 10M、边数分别为 22M,80M 和 110M 时,QueryGT 均比 Base 快两个数量级;且对于更加稠密的前者,查询效率提升的更加明显.这是由于图压缩算法对于稠密图会取得更好的压缩效果,从而令查询效率得到更大提升.其次,当查询顶点集大小$|Q|$从 5 增加至 10 时,两种算法的查询时间均有微小的增加.这是因为随着查询顶点集的增大,查询结果将有可能变成核值更小、覆盖顶点更多的连通区域,故查询时间会随之增加.



(a) 顶点数 2M,边数 22M (b) 顶点数 5M,边数 80M (c) 顶点数 10M,边数 110M

Fig.4 Query time of Base vs. QueryGT on synthetic datasets

图 4 虚拟图数据上查询算法 Base 和 QueryGT 的对比

综上所述,本文所提出的图压缩算法无论在真实还是虚拟数据集上都能取得非常可观的压缩效果,且当原始图变得稠密时,算法的压缩效果更加显著.而基于压缩图的查询性能也得到了巨大的提升.

5 结束语

本文研究了基于图压缩技术的 k-MSC 查询处理算法,提出了图压缩算法 SC 及在查询转换算法,证明了基于图压缩算法 SC 的查询的正确性.考虑到 k-MSC 查询仅需要找到符合要求的连通区域,本文提出了图压缩算法 TC 将 SC 算法得到的压缩图进一步压缩为树.本文证明了基于压缩树的查询的正确性,并给出基于压缩树的无需解压缩的查询算法.通过真实和虚拟数据上的实验结果表明:所提出图压缩算法的压缩比平均可达到 12%;而对于稠密图,算法将取得更好的接近 10%的压缩比.基于压缩算法的查询效率也得到了很好的提升,与直接在原始图上查询的 Base 算法相比,查询效率提高了 1~2 个数量级.在今后的工作中,我们将进一步探讨针对本文压缩方法的更新技术.

致谢 在此,我们向曾经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] Smith C. By the numbers: 98 amazing facebook statistics. DMR, 2014.

[2] Agrawal R, Rajagopalan S, Srikant R, Xu Y. Mining newsgroups using networks arising from social behavior. In: Proc. of the WWW 2003. 2003.

[3] Lappas T, Liu K, Terzi E. Finding a team of experts in social networks. In: Proc. of the KDD 2009. 2009.

[4] Yan X, Zhou XJ, Han J. Mining closed relational graphs with connectivity constraints. In: Proc. of the KDD 2005. 2005.

[5] Asthana S, King OD, Gibbons FD, Roth FP. Predicting protein complexmembership using probabilistic network reliability. Genome Research, 2004,14(6):1170−1175.

[6] Hanneman RA, Riddle M. Introduction to Social Network Methods. University of California, Riverside, 2005.

[7] Seidman SB. Network structure and minimum degre. Social Networks, 1983,5:269−287.

[8] Bollobas B. The evolution of sparse graphs, in graph theory and combinatorics. In: Proc. of the Cambridge Combinatorial Conf. in Honor of Paul Erdos. Academic Press, 1984. 35−57.

[9] Kortsarz G, Peleg D. Generating sparse 2-spanners. Journal of Algorithms, 1994,17(2):222−236.

[10] Andersen R, Chellapilla K. Finding dense subgraphs with size bounds. In: Proc. of the WAW. 2009. 25−37.

[11] Batagelj V, Zaversnik M. An $o(m)$ algorithm for cores decomposition of networks. Computer Science, 2003,1(6):34−37.

[12] Boldi P, Vigna S. The webgraph framework i: Compression techniques. In: Proc. of the 13th Int'l Conf. on World Wide Web. ACM Press, 2004. 595−602.

[13] Chierichetti F, Kumar R, Lattanzi S, Mitzenmacher M, Panconesi A, Raghavan P. On compressing social networks. In: Proc. of the 15th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2009. 219−228.

[14] Maserrat H, Pei J. Neighbor query friendly compression of social networks. In: Proc. of the 16th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2010. 533−542.

[15] Fan W, Li J, Wang X, Wu Y. Query preserving graph compression. In: Proc. of the 2012 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2012. 157−168.

[16] Buneman P, Grohe M, Koch C. Path queries on compressed XML. In: Proc. of the 29th Int'l Conf. on Very Large Data Bases—Vol.29. In: Proc. of the VLDB Endowment. 2003. 141−152.

[17] Navlakha S, Rastogi R, Shrivastava N. Graph summarization with bounded error. In: Proc. of the 2008 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2008. 419−432.

[18] Newman ME, Girvan M. Finding and evaluating community structure in networks. Physical Review E, 2004,69(2):026113.

[19] Huang X, Cheng H, Qin L, Tian W, Yu JX. Querying k-truss community in large and dynamic graphs. In: Proc. of the SIGMOD 2014. 2014.

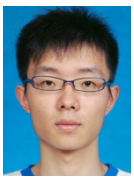
[20] Cui W, Xiao Y, Wang H, Lu Y, Wang W. Online search of overlapping communities. In: Proc. of the SIGMOD 2013. 2013.

[21] Chang L, Lin X, Qin L, Yu JX, Zhang W. Index-Based optimal algorithms for computing steiner components with maximum connectivity. In: Proc. of the 2015 ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2015. 459−474.

[22] Gibbons A. Algorithmic Graph Theory. Cambridge University Press, 1985.

[23] Aho AV, Hopcroft JE, Ullman JD. On finding lowest common ancestors in trees. In: Proc. of the STOC'73. 1973.

[24] Bollobás B, Riordan O. The diameter of a scale-free random graph. Combinatorica, 2004,24(1):5−34.



李鸣鹏(1989－),男,黑龙江勃利人,硕士,主要研究领域为图数据的查询处理.

邹兆年(1979－),男,博士,讲师,CCF 会员,主要研究领域为图数据挖掘.

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.