软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2014,25(Suppl.(2)):178−188

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

# 基于压缩的大规模图的割点求解算法*

李发明, 李建中, 邹兆年, 张冠男

(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: 李发明, E-mail: lifaming2015@126.com

**摘 要**: 割点求解是图应用中的一个重要操作.深度优先搜索树算法可以解决割点求解问题.但是该算法存在缺点,导致它不能在实际问题中得到很好的应用.这是因为当今数据的两大特点,一是数据规模庞大,对于很多图操作提出了挑战性的要求;二是数据多变,每天数据的大量更新使得传统算法必须依据更新重复计算,浪费了时间和空间.深度优先搜索树算法的时间复杂度为 $O(|V|+|E|)$,其中,$|V|$和$|E|$分别为图的顶点的数目和边的数目.它能够很好地适应第 1 个特点,但是对于第 2 个特点该算法则无能为力.提出一种基于压缩的割点求解算法来解决这个问题.该算法通过点的朴素相似来压缩图,时间复杂度为 $O(|E|)$.在得到的无损压缩图上进行割点求解,同时在压缩图上动态地维护点和边的更新,在不解压图的情况下完成图的更新,在更新后的图上进行割点求解,极大地降低了时间和空间消耗.该压缩算法得到的压缩图对其他图操作同样适用.

**关键词**: 割点;弱割点;深度优先搜索树;朴素相似性;压缩;动态维护

中文引用格式: 李发明,李建中,邹兆年,张冠男.基于压缩的大规模图的割点求解算法.软件学报,2014,25(Suppl.(2)):178−188. http://www.jos.org.cn/1000-9825/14036.htm

英文引用格式: Li FM, Li JZ, Zou ZN, Zhang GN. Cut-Vertex detection algorithm based on compression on big graph. Ruan Jian Xue Bao/Journal of Software, 2014,25(Suppl.(2)):178−188 (in Chinese). http://www.jos.org.cn/1000-9825/14036.htm

## Cut-Vertex Detection Algorithm Based on Compression on Big Graph

LI Fa-Ming, LI Jian-Zhong, ZOU Zhao-Nian, ZHANG Guan-Nan

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

Corresponding author: LI Fa-Ming, E-mail: lifaming2015@126.com

**Abstract**: The detection of cut vertex is an important operation on graph. The deep-first search algorithm can solve this problem. However, the algorithm has drawback which prevents it from applying in the real-world applications. This is because of the two characteristics for today's data. One is the scale of data is huge, so it is challenging for many operations on graph. Another challenge is that the data is changeable. Because of the massive updates, the traditional algorithm must compute repeatedly according to the change, wasting a lot of time and space. The time complexity of deep first search tree is $O(|V|+|E|)$, where $|V|$ and $|E|$ are number of nodes and edges of graph, so it can adapt to the first characteristic very well. But it is useless for the second characteristic. In order to solve this problem, this paper puts forward an algorithm based on compression to discovering cut vertex. The algorithm compresses the graph based on the naïve similarity on nodes. The time complexity of the algorithm is $O(|E|)$. It discovers cut vertex on the lossless compression graph. At the same time, the algorithm maintains the updates of the nodes and edges dynamically, and updates the graph without decompression. It discovers cut vertex on the compression graph after update. These methods reduce the consumption of time and space remarkably. The compressed graph obtained by the compression algorithm in the article can be applied to other graph operations.

**Key words**: cut-vertex; weak cut-vertex; deep-first search tree; naïve similar; compression; dynamic maintenance

很多实际应用问题可以抽象成图模型,实体抽象成顶点,实体间的某种关系抽象成边.解决好图问题就可以很好地解决与之对应的实际应用问题.割点在图的实际应用中有着重要的地位,割点求解也是一个传统的图操

* 基金项目: 国家自然科学基金(61133002); 国家重点基础研究发展计划(973)(2012CB316202)
收稿时间: 2014-05-07; 定稿时间: 2014-08-19

作.一个连通图如果去掉某一点及和其关联的边后,连通分量的数目大于 1,那么该点就是割点.如在无线传感器网络中,身为割点的传感器起着连接整个网络的作用,它的损坏将导致整个传感器网络失效.同样,在配电网络中,如果身为割点的传输基站不能正常工作,电力的输送将会出现问题.这就需要我们找到这些割点,并重点维护它们来保证整个网络的正常工作.同时,如果网络的功能出现问题,也可以对这些割点进行优先排查,减少了工作的负担.在道路网络的设计中,要尽量避免出现割点,因为割点处的交通会异常拥堵,使得交通事故出现的概率增加,同时避免割点式的交通路口出现,可以缓解交通压力.对于毒品网络,只要找到作为“割点”的那个中间人,就可以令整个毒品交易网瘫痪.传统的深度优先搜索树算法[1]能够解决割点求解问题,但是该算法忽略了一个重要问题,那就是实际应用中图数据是在经常改变的[2],如传感器网络会经常添加一些节点,配电网络会向新的区域进行电力输送,道路网络中有些道路会进行维修,如果针对每一个微小的变化都去在原图上重新计算,这样的代价是巨大的.

和割点求解一样,图压缩也是一个古老的图操作.随着图数据规模的爆炸式增长,人们对于图压缩的关注也日益增加.压缩后的图有很多优势,如大规模图的可视化、降低图上操作的时间、减少检索时间及存储空间等等.图压缩过程可以看做是进行顶点和边的聚类,可以分为两个方面,一方面是根据点或边的权值将相似的点放在一起,另一方面是根据操作将具有相同操作性质的点或边放在一起,它们都是通过减少点和边的数目来降低图的规模.文献[3−6]中提出了基于网页特性的压缩方法.它们都是利用网页的超链接具有相同前缀和相似类型的网页常常具有相同的邻居节点的特性进行压缩,但这种方法只对网络图有效.文献[7−9]在标记图上进行压缩,它们采用类似聚类的方法将图压缩,根据顶点和边的属性进行相似判断.文献[10]针对可达性和模式匹配两种操作寻找相似的点将图压缩.

衡量顶点相似性的标准通常为最小跳跃距离(min-hop distance)、Jaccard 系数[11]和 Cosine 相似[12]等等,但这些判断相似方法都存在一些问题.比如,复杂度过高,导致在大数据图上根本无法应用,如文献[9]中的算法,又如,没有保存全部边的信息,为有损压缩,如文献[13]中得到的压缩图,使得很多操作不能准确、完整地返回全部解集.文献[14,15]中的压缩只是一种聚类方法,将标号相似的点放到了一起,不能在其上进行图操作.文献[16]给出了基于最小描述长度[17]的方法来将图压缩,并且提出修正集的概念,但是针对割点求解并没有很好的效果.

针对上述问题,我们提出了一种在压缩图上进行割点求解的算法,同时在压缩图上进行增量维护,以减少每次在原始图上重新计算带来的代价.对于图中点的相似性,如果两个顶点是邻接的,它们在某种程度上就是相似的,与此同时,如果两个邻接点的度还很接近,则更加保证了这种相似.基于以上观点,我们定义了朴素相似性进行点的压缩,形成超级节点,压缩过程的时间复杂度为 $O(|E|)$,其中$|E|$为图的边的数目.由于每一条边的存在与否都会影响割点的判断,所以我们使用边编码来记录边的信息,实现无损压缩.在现有文献中还没有见到将边编码技术应用到图压缩以及在压缩图上根据边编码进行相关操作的具体方法.我们的算法得到的压缩图对其他图操作同样适用.针对割点求解,我们改进深度优先搜索树算法,使之适用于压缩图上的割点求解,保证在压缩图上求解割点的时间大大小于原始图上求解割点的时间.针对图的维护,即边和点的添加和删除,我们提出了增量式的算法,以保证在不解压的情况下实现上述操作,减少时间复杂度.实验结果表明,我们的算法有很好的效果.在压缩方面,我们得到了很好的压缩比,平均在 40%左右;在图的更新与维护方面,我们的算法的时间消耗比深度优先搜索树少很多,对于稀疏图和稠密图,效果尤为明显.

本文第 1 节给出相关的基本概念和本文研究的问题的定义.第 2 节介绍我们提出的压缩算法及改进的割点求解算法.第 3 节给出在压缩图上进行边和点的添加和删除的维护算法,同时给出我们针对批量更新的处理办法.第 4 节给出相关的实验设计和结果分析.第 5 节进行总结,概括本文的内容.

## 1 基本概念及问题定义

### 1.1 割点和弱割点

**定义 1**(**割点**). 无向图 $G=(V,E)$为连通图,对于任意的点 $u\in V$,$u$ 为割点当且仅当移除 $u$ 点以及与其关联的边后,存在某两个点 $x,y\in V$ 是不连通的.

割点的定义存在多种形式,比如,割点的删除会增加连通分量的数目,又如,任何两点之间的最短路径都经过割点等等.

图 1 为一个无线传感器网络抽象图.图中存在 4 个割点,分别为点 $a$、点 $b$、点 $i$ 和点 $j$.如果点 $a$ 或点 $b$ 发生通信故障,则整个网络的通信将瘫痪,传感器网络将失去效用.如果点 $j$ 出现问题,点 $l$ 和点 $m$ 将从原图中分离出去.至于点 $i$,我们将采用下面的定义.

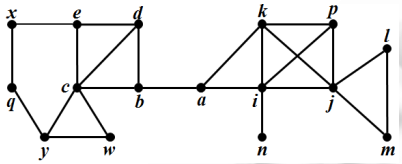


Fig.1 Wireless sensor network

图 1 无线传感器网络

**定义 2(弱割点)**. 无向图 $G=(V,E)$为连通图,若点 $u\in V$ 为割点,同时点 $u$ 只是将一个点 $v\in V$ 从连通图中分离出去,则称点 $u$ 为弱割点.

如图 1 中的点 $i$ 为弱割点,点 $i$ 的删除只是将点 $n$ 从连通图中去除,并不会影响大部分其他节点之间的通信.我们的算法会将割点和弱割点分开返回,以便根据实际应用来区分点的重要性.图 1 中的点 $j$ 也只是将点 $l$ 和点 $m$ 从原图中分离出来,但是这里并不将其定义为弱割点,因为点 $l$ 和点 $m$ 之间仍然可以保持通信.

**定义 3(弱割点性)**. 无向图 $G=(V,E)$,任意的点 $u\in V$,若点 $u$ 为弱割点,则称点 $u$ 具有弱割点性.

一个点具有弱割点性,说明该点的邻接点中一定有一个度为 1 的点,在压缩算法预处理过程中一定会将这个度为 1 的点删除,同时记录它所对应的弱割点.如图 1 中点 $i$ 具有弱割点性,在压缩算法的预处理过程中点 $n$ 将会被删除,点 $n$ 对应的弱割点 $i$ 将会被记录下来.

### 1.2 压缩和朴素相似性

**定义 4(图压缩)**. 无向图 $G=(V_G,E_G)$,$V_G$ 可以分解为 $V_{G_1}\cup V_{G_2}\cup\ldots\cup V_{G_k}$,同时,$V_{G_i}\cap V_{G_j}$ $(1\leqslant i\neq j\leqslant k)$,则压缩图 $G_S=(V_S,E_S)$ 满足如下定义:$V_S$ 有 $k$ 个顶点 $V_{G_1},V_{G_2},\ldots,V_{G_k}$ 对应于 $(V_{G_i}\to V_i)$,边 $(V_i,V_j)\in E_S$ 代表 $V_i$ 和 $V_j$ 之间的边.

图 2 为我们的压缩算法在图 1 上运行后得到的压缩图.其中,两个原始图中的顶点合并为一个超级节点,如最左边的两个点可表示为 $V_{qx}$ 和 $V_{wy}$,超级节点中的子节点依照字典序(如果为数字,则按数字升序由小到大)排列,这是为后面边编码做准备.同时,与弱割点相连的度为 1 的点被删除,因为它们已经确定了弱割点的存在.

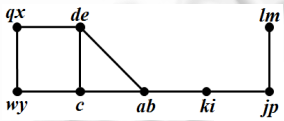


Fig.2 Effect of compression

图 2 压缩效果图

我们的压缩依据是顶点相似性.与文献[13,14]中提到的相似性不同,本文的相似性是一种朴素的顶点相似性,其定义如下(其中,$\mu(v)$表示顶点 $v$ 的度):

**定义 5(朴素相似)**. 图 $G=(V,E)$,$v$ 及 $v_1,v_2,\ldots,v_k\in V$,且 $v$ 和 $u_1,u_2,\ldots,u_k$ 邻接,对于 $u_1,u_2,\ldots,u_k$ 中的任意一点 $u$,若 $|\mu(v)-\mu(u)|=\min|\mu(v)-\mu(v_i)|$,其中 $1\leqslant i\leqslant k$,则称 $v$ 和 $u$ 为朴素相似.

本文提出的朴素相似是基于邻接关系的相似,即邻接就可以看成一种相似.如在社交网络中两个人为好友关系,则他们一定在某一方面或多个方面存在相似,如爱好、作息时间、共同朋友等等,而当这两个好友的在社交网络中的朋友数目相近时,说明他们的习惯或性格更加相似.又如,在无线传感器网络中,相互通信的传感器

节点一定具有相似之处,如相同的通信协议、相似的监测功能等等.文中定义的朴素相似保证压缩过程中的相似性判断的复杂度为 $O(|E|)$,即整个压缩过程的时间复杂度为 $O(|E|)$,其中,$|E|$为图的边的数目,这保证了该算法可以很好地处理大规模数据图.

### 1.3 边编码

由于任何一条边的存在与否都会影响割点的判定,所以我们的压缩方法必须是无损压缩.于是,我们就需要记录两个超级节点间内部节点之间的邻接关系.针对这个问题,我们提出了边编码的概念.超级节点间的边编码定义如下:

**定义 6(边编码)**. 无向图 $G=(V,E)$及压缩图 $G_S=(V_S,E_S)$ 的任何两个超级节点 $V_{uv},V_{pq}\in V_S$, 设置 4 位 bit 记录超级节点中子节点邻接关系,若超级点中的两个子节点 $up\in E$,则在第 1 个 bit 处记录 1,否则记录 0,以此类推,编码剩余超级节点中子节点.我们称这种编码方式为边编码.

对于超级节点和非超级节点间的编码只需要两位 bit 来记录邻接关系,编码方式与定义 6 一样.

图 3 为压缩后图上的边编码.

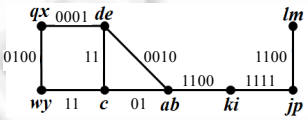


Fig.3 Edge code of compressed graph

图 3 压缩图边编码

本文的研究内容是利用朴素相似压缩图,在得到的带边编码的压缩图上求解割点,返回所有满足割点定义的点,包括弱割点.同时针对这种压缩图,我们还会进行增量维护,实现在不解压压缩图的情况下,完成图的更新,并返回更新后的图的全部割点.

## 2 压缩算法和压缩图上的割点求解算法

本节我们将提出基于朴素相似的图压缩算法以及压缩图上的割点求解算法.我们先给出具体的算法,然后就算法给出时间复杂度分析.

### 2.1 朴素相似压缩算法

**算法 1**. Naïve_Similar_Compress(朴素相似压缩).

输入:无向图 $G=(V,E)$.

输出:压缩图 $G_S=(V_S,E_S)$,弱割点集合 $V_W$.

```
1.  for (i=0; i<|V|; i++)
2.      if (v_i 满足弱割点性)
3.          删除与 v_i 邻接的度为 1 的点,并将 v_i 加入到 V_W 中;
4.  for (i=0; i<|V|; i++)
5.      if (v_i 没有被压缩)
6.          if (存在 v_i 的朴素相似点 u)
7.              将 v_i 和 u 从原图中分裂出来,合并为一个超级节点加入到超级节点集 V_S 中;
8.              将 v_i 和 u 做压缩过标记;
9.          else
10.             将 v_i 从原图中分裂出来,加入到超级节点集合 V_S 中,将 v_i 做压缩过标记;
11. for (每一超级节点)
```

12. if (超级节点的内部节点间存在邻接关系)
13. 在超级节点间添加边,同时添加边编码;
14. 返回图 $G_S=(V_S,E_S)$及弱割点集 $V_W$;
15. 算法结束;

Naïve_Similar_Compress(朴素相似压缩)首先进行剪枝操作,同时将弱割点返回(第 1 步~第 3 步).这种处理方法对稀疏图尤为有效,能够将图中的单个点、单条路径等直接修剪掉,极大地减少了下面割点求解算法的运算时间.(第 3 步~第 10 步)进行分割操作,找到点的朴素相似点,并从原图中分离出去,同时记录点已经被压缩,避免重复压缩.(第 11 步~第 14 步)进行压缩图的构造,通过内部顶点间的邻接关系连接超级节点,同时对超级节点之间的边进行编码,保存原图中每一个顶点对之间的邻接关系.至此,我们得到了压缩图,同时保证这种压缩为无损压缩,即可以由压缩图无损地还原出原始图.由于压缩过程只需对每条边验证一次即可,故压缩过程的时间复杂度为 $O(|E|)$,其中,$|E|$为图的边的数目.这样的时间复杂度保证了本文的压缩算法可以很好地应用到大规模数据图上.

### 2.2 压缩图上的深度优先搜索树算法

我们的算法是在深度优先搜索树算法的基础上进行改进,以适用压缩算法得到的压缩图,具体算法如下:

**算法 2**. DFS_For_Cut_On_Compressed_Graph.

输入:压缩图 $G_S=(V_S,E_S)$.

输出:割点集 $V_C$.

```
1.  int preNum[|V_S'|];
2.  int backNum[|V_S'|];
3.  int index=0;
4.  for (每一个 v∈V_S)
5.  if (v 没有被访问过)
6.  DFS_Cut(G_S,v,v,intdex,preNum,backNum)
Procedure DFS_Cut(G,v,root,intdex,preNum,backNum)
1.  v 做访问过标记;
2.  preNum[v]=++ index;
3.  backNum[v]=preNum[v];
4.  for (v 的每一个邻接顶点 u)
5.      if (u 没有被访问过)
6.          DFS_CUT(G,u,root,index,preOrder,backOrder);
7.          if (v 为根且子节点数为 2,并且去掉 v 中任意子节点 v_i 后,编码中不存在 1)
8.              v_i 加入到 V_C 中;
9.          else
10.             backNum[v]更新为 v 可到达的最小祖先节点;
11.             if (backNum[u]≥preNumber[v]且去掉 v 子节点 v_i 后,uv 对应的边编码不存在 1)
12.                 v_i 加入到 V_C 中;
13.             else
14.                 backNum[v]更新为 v 可达的最小祖先节点;
15. 返回割点集 V_C;
16. 算法结束;
```

该算法采用 $preNum[v]$和 $backNum[v]$分别记录任意一个点 $v$ 的访问顺序和可以到达的最小祖先节点,通过比较 $v$ 和 $v$ 的后继节点可以到达的最小祖先节点的大小确定 $v$ 是否为割点,同时利用边编码确定每一个超级节

点内的节点是否为割点,由于采用的是 0、1 序列的位运算,故时间消耗非常小.该算法的时间复杂度为 $O(|V|+|E|)$[1],其中,$|V|$和$|E|$分别为压缩图的顶点的数目和边的数目.上述算法采用递归实现深度优先遍历,但数据规模变大后,需将递归算法转化成栈实现,因为递归深度过大时,系统分配的栈会溢出.

## 3 增量维护

在任何实际应用中都会存在更新问题,如在无线传感器网络中会添加新的传感器和删除原有的传感器,会在传感器间增加通信和删除通信,对应到图上就是顶点的添加和删除操作及边的添加和删除操作.所以必须将更新考虑到算法中,避免每一次的更新都需要在原图上进行维护,并且重新运行算法,这样才能降低每次更新给算法带来的运算代价.如果针对每一次的更新都要在原始图上重新计算,就会给计算机带来极大的负担.

下面我们分 3 个部分提出维护算法,分别为边的更新、点的更新和批量更新.

### 3.1 边的删除和添加

边的添加是指在已有的顶点间添加一条新的边,对应到实际应用中就是实体点间添加了某种关系,如增加通信、建立好友关系等等,具体算法如下:

**算法 3**. Add_Edge.

输入:压缩图 $G_S=(V_S,E_S)$,弱割点集 $V_W$,添加边 $uv$.

输出:更新后的压缩图 $G_S'=(V_S',E_S')$.

1. if ($u$ 和 $v$ 都是超级节点的子节点)
2. 查询添加边 $uv$ 端点所在超级节点 $V_{S_1}$ 和 $V_{S_2}$;
3. if ($V_{S_1}$ 和 $V_{S_2}$ 邻接)
4. 将相应位置边编码置 1;
5. else
6. 将边 $V_{S_1}V_{S_2}$ 添加到 $E_S$ 中,同时进行边编码;
7. else if ($u$ 为超级节点 $V_U$ 的子节点,$v$ 为弱割点分离出的点)
8. 将 $v$ 加入到新的超级节点 $V_0$ 中,$V_0$ 添加到 $V_S$ 中,将边 $V_UV_0$ 添加到 $E_S$ 中;
9. 由 $V_W$ 中删除 $v$ 对应的弱割点;
10. else
11. 将 $uv$ 合并为超级节点 $V_{UV}$,将其添加到 $V_S$ 中,将邻接边添加到 $E_S$ 中;
12. 由 $V_W$ 中删除 $u,v$ 对应弱割点;
13. 返回新的压缩图 $G_S'=(V_S',E_S')$;
14. 算法结束;

在压缩图上添加的边分为 3 种类型,即邻接的两个端点都在超级节点中、一个在超级节点中和都不在超级节点中.第 1 种类型只需在压缩图中直接添加边即可,后两种类型则需要将不在压缩图中的点以及新添加的边加入到压缩图中.这种操作的时间复杂度为常数时间.

边的删除是指删除原图中已有的边,对应到实际应用中就是实体点间删除了某种关系,如删除通信、解除好友关系等等.具体算法如下:

**算法 4**. Delete_Edge.

输入:压缩图 $G_S=(V_S,E_S)$,弱割点集 $V_W$,删除边 $uv$.

输出:更新后的压缩图 $G_S'=(V_S',E_S')$.

1. if ($u$ 和 $v$ 都是超级节点子节点)
2. 查询删除的边 $uv$ 端点所在超级节点 $V_{S_1}$ 和 $V_{S_2}$,将相应位置边编码置 0;
3. if ($V_{S_1}$ 和 $V_{S_2}$ 的边编码为 0000)

4. 删除边 $V_{S_1}V_{S_2}$;
5. else if ($u$ 为超级节点 $V_U$ 的子节点,$v$ 为弱割点分离出的点)
6. 从 $V_W$ 中删除 $v$ 对应的弱割点;
7. else
8. 删除 $u$,$v$ 对应的弱割点;
9. 返回新的压缩图 $G_S'=(V_S',E_S')$;
10. 算法结束;

在压缩图上删除的边同样分为 3 种类型,即邻接的两个端点都在超级节点中、一个在超级节点中和都不在超级节点中.第 1 种情况只需将边的两个端点所对应的超级节点找到,将相应位置边编码置 0,如果超级节点间边编码全部为 0,则删除原有超级节点之间的边.对于第 2 种情况,只需删除点 $v$ 所对应的弱割点即可,因为点 $v$ 和弱割点之间不再存在边.第 3 种情况只需删除点 $u$ 和点 $v$ 确定的弱割点即可.这种操作的时间复杂度为常数时间.

### 3.2 顶点的添加和删除

顶点的添加和删除是对边的添加和删除的扩展,因为在图模型中,点是与边关联的.

点的添加是指原图中添加新的点,对应到实际应用中就是添加新的实体,如在无线传感器网络中添加新的节点并与已存在的传感器进行通信、注册社交网络账号并添加好友等等.具体算法如下:

**算法 5**. ADD_Node.

输入:压缩图 $G_S=(V_S,E_S)$,添加点 $v$ 及边 $vu_0,vu_1,...,vu_n$, 弱割点集 $V_W$.

输出:更新后的压缩图 $G_S'=(V_S',E_S')$.

1. 将 $v$ 添加到新的超级节点 $V_V$ 中;
2. for (每一条边 $vu_i$)
3. 调用 ADD_Edge;
4. 返回新的压缩图 $G_S'=(V_S',E_S')$;
5. 算法结束;

我们只需将 $v$ 加入到 $V_S$ 中即可,同时调用上节的添加边(Add_Edge)算法来处理与点 $v$ 邻接的每一条边.这种操作的时间复杂度为 $O(N)$,其中,$N$ 为添加的与点 $v$ 邻接的边的数目.

点的删除是指删除原图中已有的点,对应到实际应用中就是实体点的删除,如去掉无线传感器网络中的某个节点、注销社交网络账号等等.具体算法如下:

**算法 6**. Delete_Node.

输入:压缩图 $G_S=(V_S,E_S)$,弱割点集 $V_W$,删除点 $v$.

输出:更新后的压缩图 $G_S'=(V_S',E_S')$.

1. if ($v$ 在弱割点集 $V_W$ 中)
2. 将 $v$ 从 $V_W$ 删除;
3. 算法结束;
4. 查询 $v$ 所在超级节点 $V_V$;
5. if ($V_V$ 包含一个子节点)
6. 将 $V_V$ 做删除标记;
7. else
8. 将 $v$ 在 $V_V$ 中删除;
9. 将 $v$ 对应的编码置 0;
10. for (每一条 $V_V$ 邻接的边)

11. if (边编码进行或操作结果为 0)
12. 删除相应边;
13. 返回新的压缩图 $G_S'=(V_S',E_S')$;
14. 算法结束;

删除点 $v$ 后,与点 $v$ 邻接的边也将被删除,点 $v$ 可能在两个集合中,弱割点集或超级节点集.在弱割集中直接删除即可.在超级节点集中需要进行边编码的修改以及超级节点之间是否存在边的判断,确定超级节点之间的边是否需要保留.这种操作的时间复杂度为 $O(N)$,其中,$N$ 为删除的与点 $v$ 邻接的边的数目.

### 3.3 批量添加边和点

有些时候,实际应用中会添加大量的边和点,如在无线传感器网络应用中,新增一个无线传感器群来监测一片新的区域,并且与已经存在的传感器网络进行通信.又如,在配电网络中,向一片新建的区域配送电力.针对上述情况,我们需要更新原有压缩图,将新的区域更新添加到原有压缩图中.本文采用的批量更新方法仍是在原有的压缩图上进行更新,先将新的更新的图压缩,然后将得到的压缩图添加到原来的压缩图上.具体的批量更新算法如下.

**算法 7**. Batch_Update.

输入:压缩图 $G_S=(V_S,E_S)$,添加图 $G=(V,E)$.

输出:更新后的压缩图 $G_S'=(V_S',E_S')$.

1. 对于图 $G$,调用算法 Naïve_Similar_Compress,得到压缩图 $G_S''=(V_S'',E_S'')$;
2. for (每一个 $V_{S_1}\in V_S''$)
3. 调用 ADD_Node 算法;
4. 返回新的压缩图 $G_S'=(V_S',E_S')$;
5. 算法结束;

调用 Naïve_Similar_Compress 算法将添加图进行压缩,然后调用 ADD_Node 算法将得到的压缩图与原来的压缩图进行连接,同时将超级节点加入到 $V_S$ 中,将它们之间的边加入到 $E_S$ 中.这种操作的时间复杂度为 $O(|E|)$,其中,$|E|$为新添加图的边的数目.

## 4 实验结果

我们从两个方面验证我们算法的效率,包括压缩算法得到的压缩比及针对更新传统深度优先搜索树算法运行时间与提出的割点求解算法运行时间比,主要体现我们的算法在图压缩和更新维护方面的优势.

实验采用的计算机主频为 3.1GHZ,内存 4G.我们的全部算法均在 win8 操作系统、eclipse 编译器上采用 java 实现,由于只需要考虑比值(压缩比和算法时间消耗比),所以 java 的效率问题不会对我们的结果产生较大的影响.我们针对每个数据做 5 次实验,取 5 次实验结果的平均值绘制图像.

实验数据中的真实数据全部来自斯坦福 SNAP 数据库,虚拟数据采用 R-MAT 模拟实现.数据细节见表 1.

**Table 1** Scale of graph data

**表 1** 图数据规模

| ID | Dataset | #Vertices | #Edges |
|---|---|---|---|
| D1 | R-MAT | 2.048 | 5.530 |
| D2 | ca-GrQc | 5.242 | 14.484 |
| D3 | ca-CondMat | 23.133 | 93.439 |
| D4 | email-Enron | 36.692 | 367.662 |
| D5 | R-MAT | 26.2144 | 7.123.674 |
| D6 | roadNet-CA | 1.965.206 | 5.533.214 |
| D7 | R-MAT | 4.194.304 | 9.284.912 |

### 4.1 压缩比

图 4 中的纵轴表示压缩比.这里,我们采用 $\eta=|E_S|/|E|$来定义压缩比.对表 1 中的各图分别调用朴素相似压缩算法进行压缩操作,得到的压缩比如图 4 所示.

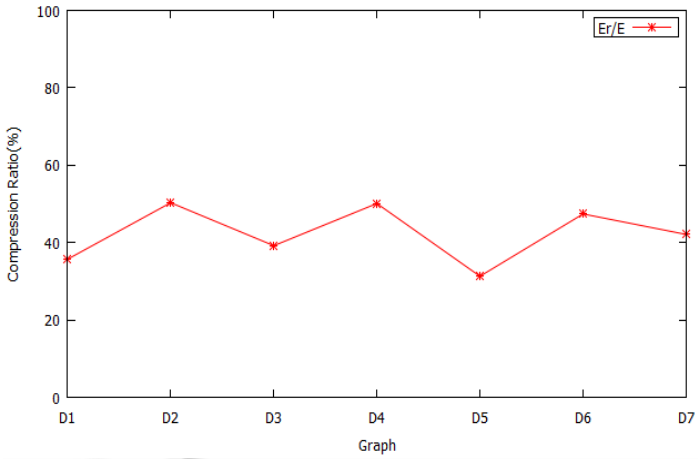


Fig.4 Ratio of compression

图 4 压缩比

由图 4 可以看出,针对不同规模的图数据得到的压缩比各不相同,但压缩比基本在 40%左右.我们的算法在处理稀疏图和稠密图时效果比较明显,因为稀疏图在剪枝过程中将会减掉一大部分与弱割点连接的点,这样就使图的规模得以减小;对于稠密图,当点合并为超级节点后,边的数目急剧减少,如大小为 4 的完全图,边的数目由 6 变为 1,点的数目变为 2,这样就使压缩比变得很大.但是,D2 和 D3 虽然为稀疏图,但是并没有得到很好的压缩比,这是因为图中的连通分量的数目过于多,导致压缩效果没有同等规模的连通分量数目少的稀疏图的效果好,而对于采用 R-MAT 生成的稠密图 D5,我们得到了很好的压缩比,与分析的结果基本相似.同时可以看到,我们的算法可以处理大规模的数据图,如 D7 这样的图,这是大多数其他压缩算法在单机上做不到的.

### 4.2 割点求解时间比

图 5 中的纵轴表示针对更新本文提出的算法求解割点的时间和传统的深度优先搜索树求解割点的时间的比($T_r/T$).

由图 5 中我们看到,算法在针对更新时,求解割点的时间有很大的提高,这是因为压缩后的图的点的数目和边的数目都减少了许多,并且在压缩图上维护更新的时间相对较少.而稀疏图更是因为提前处理弱割点而得到很好的结果.对于稠密图,由于压缩效果好而大量减少了点和边的数目,使得输入规模变小而减少运算时间.虽然更新有 4 种类型,但是它们在进行更新时,处理耗时非常少,可以看作是等价的操作,故 4 种操作的消耗的时间可以看作相同的,在实验图上可以采用同一点表示.对于批量更新,更新原图与更新压缩图消耗的时间比值基本固定,故得到的求解割点时间比例和图 5 仍然基本一致,同样采用如图 5 所示的曲线表示.综上所述,对于 5 种操作的实验曲线重合为一条曲线.通过比较发现,图 4 和图 5 曲线形状基本一样,这是因为改进的深度优先搜索树的时间复杂度为 $O(|V|+|E|)$,与点和边的数目成正相关.

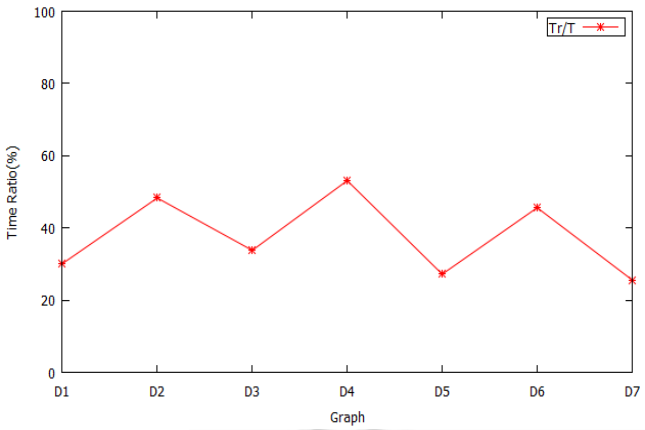


Fig.5 Time ratio for cut-detection

图 5 针对更新求解割点时间比

## 5 结束语

本文提出了基于朴素相似性的压缩方法,使用边编码的方式记录原图中点的邻接关系,保证压缩过程为无损压缩,并且在压缩图上采用改进的深度优先搜索树的方法来求解割点,同时可以在压缩图上实现动态增量维护,保证在不解压原图的基础上来维护图的更新,极大地减小了传统算法针对每次更新都需要在原始图上重新计算的时间代价.压缩算法的时间复杂度为 $O(|E|)$,可以很好地应用到大规模数据集上.通过在真实数据图和虚拟图上的实验证明本文的压缩算法能够得到较好的压缩比.针对割点求解这一操作,相对于传统深度优先搜索树算法,我们的算法能够更好地适应实际应用图的多变性.本文得到的压缩图适用于其他类型的图操作,同样能够节省计算时间,并且可以很好地处理大规模数据.接下来,我们将继续研究将多个顶点合并为一个超级节点时所需要的边编码技术、检索方法及维护算法,以便得到更好的压缩比和更快的割点求解时间.

**References**:

[1] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 2nd ed., The MIT Press, 2002.

[2] Kumar R, Novak J, Tomkins A. Structure and evolution of online social networks. In: Link Mining: Models, Algorithms, and Applications. New York: Springer-Verlag, 2010. 337−357.

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

[4] Raghavan S, Garcia-Molina H. Representing Web graphs. In: Proc. of the 19th Int'l Conf. on Data Engineering. IEEE, 2003. 405−416.

[5] Adler M, Mitzenmacher M. Towards compressing Web graphs. In: Proc. of the Data Compression Conf. (DCC 2001). IEEE, 2001. 203−212.

[6] Apostolico A, Drovandi G. Graph compression by BFS. Algorithms, 2009,2(3):1031−1044.

[7] Zhang N, Tian Y, Patel JM. Discovery-Driven graph summarization. In: Proc. of the 2010 IEEE 26th Int'l Conf. on Data Engineering (ICDE). IEEE, 2010. 880−891.

[8] Tian Y, Hankins RA, Patel JM. Efficient aggregation for graph summarization. In: Proc. of the 2008 ACM SIGMOD Int'l Conf. on Management of Data. ACM, 2008. 567−580.

[9] Toivonen H, Zhou F, Hartikainen A, Hinkka A. Compression of weighted graphs. In: Proc. of the 17th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM, 2011. 965−973.

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

[11] Dubes RC, Jain AK. Algorithms for Clustering Data. Prentice Hall, 1988.

[12] Salton G, McGill MJ. Introduction to Modern Information Retrieval. McGraw-Hill Book Company, 1983.

[13] Clauset A, Newman ME J, Moore C. Finding community structure in very large networks. Physical review E, 2004,70(6):066111.

[14] Zhou Y, Cheng H, Yu JX. Graph clustering based on structural/attribute similarities. Proc. of the VLDB Endowment, 2009,2(1): 718−729.

[15] Gilbert AC, Levchenko K. Compressing network graphs. In: Proc. of the LinkKDD Workshop at the 10th ACM Conf. on KDD. 2004.

[16] 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, 2008. 419−432.

[17] Rissanen J. Modeling by shortest data description. Automatica, 1978,14(5):465−471.

**李发明**(1988－),男,黑龙江齐齐哈尔人,硕士生,主要研究领域为海量图数据处理.
E-mail: lifaming2015@126.com

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

**李建中**(1950－),男,教授,博士生导师,主要研究领域为海量数据管理与计算,无线传感器网络,CPS.
E-mail: lijzh@hit.edu.cn

**张冠男**(1989－),男,硕士生,主要研究领域为空间数据库.
E-mail: achaly@163.com