软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2018,29(3):627−641 [doi: 10.13328/j.cnki.jos.005456]

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

# 基于 MapReduce 的图结构聚类算法*

张伟鹏[1], 李振军[1], 李荣华[1], 刘宇鸿[1], 毛 睿[1], 乔少杰[2]

[1](深圳大学 计算机与软件学院,广东 深圳 518060)

[2](成都信息工程大学 网络空间安全学院,四川 成都 610225)

通讯作者: 李振军, E-mail:15323940@qq.com

**摘 要**: 图结构聚类(SCAN)是一种著名的基于密度的图聚类算法,该算法不仅能够找到图中的聚类结构,而且还能发现图中的 Hub 节点和离群节点.然而,随着图数据规模越来越大,传统的 SCAN 算法的复杂度为 $O(m^{1.5})$($m$ 为图中边的条数),因此很难处理大规模的图数据.为了解决 SCAN 算法的可扩展性问题,提出一种基于 MapReduce 的海量图结构聚类算法 MRSCAN,这是一种计算核心节点以及两种合并聚类的 MapReduce 算法.最后,在多个真实的大规模图数据集上进行实验测试,实验结果验证了算法的准确性、有效性以及可扩展性.

**关键词**: 图数据;并行计算模型;MapReduce;图结构聚类

**中图法分类号**: TP311

中文引用格式: 张伟鹏,李振军,李荣华,刘宇鸿,毛睿,乔少杰.基于 MapReduce 的图结构聚类算法.软件学报,2018,29(3):627−641. http://www.jos.org.cn/1000-9825/5456.htm

英文引用格式: Zhang WP, Li ZJ, Li RH, Liu YH, Mao R, Qiao SJ. MapReduce-Based graph structural clustering algorithm. Ruan Jian Xue Bao/Journal of Software, 2018,29(3):627−641 (in Chinese). http://www.jos.org.cn/1000-9825/5456.htm

## MapReduce-Based Graph Structural Clustering Algorithm

ZHANG Wei-Peng[1], LI Zhen-Jun[1], LI Rong-Hua[1], LIU Yu-Hong[1], MAO Rui[1], QIAO Shao-Jie[2]

[1](College of Computer Science & Software Engineering, Shenzhen University, Shenzhen 518060, China)

[2](School of Cybersecurity, Chengdu University of Information Technology, Chengdu 610225, China)

**Abstract**: Graph Clustering is a fundamental task for graph mining which has been widely used in social network analysis related applications. Graph structural clustering (SCAN) is a well-known density-based graph clustering algorithm. SCAN algorithm can not only find the clusters in a graph, but also be able to identify hub nodes and outliers. However, with the growing graph size, the traditional SCAN algorithm is very hard to handle massive graph data, as its time complexity is $O(m^{1.5})$ ($m$ is the number of edges in the graph). To overcome the scalability issue of SCAN algorithm, this paper proposes a MapReduce based graph structural clustering algorithm, called MRSCAN. Specifically, the paper develops a MapReduce based similarity computation, a core node computation, as well as two clustering merging algorithms. In addition, it conducts extensive experiments over serval real-world graph datasets, and results demonstrate the accuracy, effectiveness, and scalability of the presented algorithm.

**Key words**: graph data; parallel computing model; MapReduce; structural graph clustering

* 基金项目: 国家自然科学基金(61402292, 61772091); 国家自然科学基金广东省联合基金(U1301252); 教育部人文社会科学研究规划基金(15YJAZH058)

Foundation item: National Natural Science Foundation of China (61402292, 61772091); National Natural Science Foundation of China Guangdong Joint Fund Project (U1301252); Planning Foundation for Humanities and Social Sciences of Ministry of Education of China (15YJAZH058)

本文由基于图结构的大数据分析与管理技术专刊特约编辑林学民教授、杜小勇教授、李翠平教授推荐.

收稿时间: 2017-08-03; 修改时间: 2017-09-05; 采用时间: 2017-11-07; jos 在线出版时间: 2017-12-05

CNKI 网络优先出版: 2017-12-06 15:37:09, http://kns.cnki.net/kcms/detail/11.2560.TP.20171206.1536.019.html

如今,随着社会的进步与科技的快速发展,大数据[1]已经变得越来越普及,逐渐成为一个耳熟能详的概念.在海量数据研究中,高性能是必须予以解决的问题之一.虽然大数据研究面临着来自各个层面的挑战,但同时也存在着机遇,如利用复杂的算法和程序从大量数据中挖掘出关键有用的信息,到利用高性能技术与系统及时获取有用的内容.当前,如何利用主流的系统如 Hadoop[2]来应对大数据应用挑战成了的一个热门的研究方向.在本文中,主要研究如何利用分布式的存储和计算架构来构建海量图数据以及处理对图数据进行数据挖掘的问题.

在图数据挖掘中,图聚类是基本的研究任务.图结构聚类(SCAN)是基于密度的聚类方法,该方法不仅能够挖掘图中的聚类结构,而且还能找到图中的 Hub 节点以及离群节点.随着图数据规模的不断扩大,现有的 SCAN 算法已经无法满足大规模图数据的需求.目前的 SCAN 算法优化都是针对串行的 SCAN 算法的剪枝优化,例如,Shiokawa 等人[3]提出了剪枝的 SCAN 算法 SCAN++,以及 Chang 等人提出了 pSCAN[4].尽管这些剪枝方法能够大幅提升 SCAN 算法的效率,但在大规模图上依然比较耗时,而且针对超大规模图数据依然无法处理.

为了解决 SCAN 算法对大数据的可扩展性问题,本文提出一种基于 MapReduce 的结构图聚类算法.具体地,首先设计了基于 MapReduce 的核心节点计算算法,然后提出了两种合并聚类的 MapReduce 算法.算法能够有效地剪枝不必要的计算,从而大幅度地减少 IO 的次数,进而提升算法的性能.最后,通过在真实的大规模图数据上进行实验验证,提出的算法呈现接近线性的可扩展性.

本文的主要贡献包括以下两个方面.

1) 提出基于 MapReduce 分布式框架的图结构聚类算法及剪枝的优化方法;
2) 通过大规模的真实图数据测试,实验结果表明,提出的两个合并聚类算法具有较好的可扩展性.

## 1 相关概念

在本节中,主要简略介绍 MapReduce 分布式计算框架设计的流程及其原理,同时引进结构聚类的相关基本概念.

### 1.1 MapReduce框架

MapReduce 是一种编程模型,是 Google 公司于 2004 年提出的能并发处理海量数据的并行编程模型[5],允许开发人员开发高度可扩展和容错的并行应用程序来处理分布式无共享环境中的大数据.MapReduce 算法在执行时,每轮涉及 3 个阶段:map,shuffle 和 reduce.假设输入数据作为一组键值对存储在分布式文件系统[6]中,3 个阶段的工作如下.

- map:在这个阶段,每个机器从分布式文件系统读取一部分键值对 $\{(k_i^m,v_i^m)\}$,并生成一组新的键值对 $\{(k_i^s,v_i^s)\}$在 shuffle 阶段[7]转移到其他机器,其本质应用在于需要对数据一对一的元素进行映射转换,通常是对数据进行截取过滤或者转化为算法需要的任何字符形式;
- shuffle:map 阶段生成的键值对 $\{(k_i^s,v_i^s)\}$在所有机器上进行 shuffle.shuffle 阶段又可以分为 Map 端的 shuffle 和 Reduce 端的 shuffle,在 shuffle 阶段结束时,保证具有相同键值 $k_i^s$ 的所有键值对 $\{(k_i^s,v_1^s),(k_i^s,v_2^s),...\}$到达同一台机器;
- reduce:每个机器将具有相同密钥 $k_i^s$ 的密钥值对组合在一起作为 $(k_i^s,\{v_1^s,v_2^s,...\})$,从一组新的键值对 $\{(k_i^r,v_j^r)\}$生成并存储在分布式文件系统中,以在下一轮中进行处理.

在每一轮中,至少需要实现两个函数:map 函数和 reduce 函数.map 函数确定如何从 $\{(k_i^m,v_j^m)\}$生成 $\{(k_i^s,v_j^s)\}$,而 reduce 函数决定如何从 $(k_i^s,\{v_1^s,v_2^s,...\})$生成 $\{(k_i^r,v_j^r)\}$.

### 1.2 结构聚类的基本概念

结构聚类算法[8]是基于密度的聚类算法改良的一种社区结构发现算法,其主要思想在于利用节点的邻居节点来作为聚类的标准.在详细描述使用 MapReduce 框架对网络图数据进行结构聚类之前,首先给出结构聚类一些基本符号的解释和定义.

**定义 1(顶点网络)**. 假设 $v$ 是图中节点,令 $v\in V$,那么节点 $v\in V$ 的结构由它和它的邻居节点所构成.用 $N(v)$ 表示 $N(v)$是一个点集,其中包含的节点元素是与点 $v$ 有边相连的节点,$(v,u)$表示以 $v$ 为顶点的边,如果该点集中不包括点 $v$,那么就称 $N(v)$是点 $v$ 的开邻居集合,即 $N(v)=\{u|(v,u)\in E\}$;若是包含点 $v$ 本身,则称为闭邻居集合,即 $N(v)=\{u|(v,u)\in E\}\cup v$.

**定义 2(结构相似性)**. 用来描述有向图中任意两个节点结构相似性的符号是$\sigma(v,u)$,表示为两个节点共同邻居数与两个节点邻居数目的几何平均数的比值(邻居数包含节点本身),其中,$N[x]$表示节点 $x$ 及其相邻节点所组成的集合,公式化为

$$\sigma(v,u)=\frac{|N[v]\cap N[u]|}{\sqrt{|N[v]\cdot N[u]|}} \tag{1}$$

**定义 3($\varepsilon$邻居)**. 给定一个节点 $v$,与 $v$ 节点满足一定的相似度的邻居,称为 $v$ 的$\varepsilon$邻居.$\varepsilon$是用于划分邻居与非邻居的相似度阈值.给定一个节点 $v$,所有满足$\sigma(v,u)\geqslant\varepsilon$的节点都是 $v$ 的$\varepsilon$邻居.若$\varepsilon$=0,则图中所有节点均互为$\varepsilon$邻居节点;若$\varepsilon\neq$0,则$\varepsilon$邻居的公式化表达是 $N_\varepsilon[v]=\{u\in N[v]|\sigma(u,v)\geqslant\varepsilon\}$.即,$u$ 满足是 $v$ 的$\varepsilon$邻居必须满足以下两个条件:

$$u\in N(v),\sigma(v,u)\geqslant\varepsilon \tag{2}$$

**定义 4(核心节点)**. 当一个节点与其足够数目的邻居均满足一定相似度的时候,该节点就是一个核心节点.核心节点 $v$ 是特殊的节点,满足有 $N_\varepsilon[v]\geqslant\mu$,即满足$\varepsilon$邻居的数目大于$\mu$,$\mu$是阈值,满足该条件的点就是核心点.直观而言,核心节点在图中稠密处并且与周围的点相似度高.在本文中,用 *isCore* 变量来标记一个节点是否为核心节点.

**定义 5(直接结构可达)**. 若节点 $u$ 是核心节点的邻居的$\varepsilon$节点,则称 $v$ 直接可达 $u$,记作:

$$DirREACH_{\varepsilon,\mu}(v,u) \tag{3}$$

若 $v$ 和 $u$ 都是核心节点,那么这种可达的关系是具有对称性的,记作:

$$DirREACH_{\varepsilon,\mu}(v,u)\Leftrightarrow DirREACH_{\varepsilon,\mu}(u,v) \tag{4}$$

若只有 $v$ 是核心节点,那么这种可达关系是不具有对称性的,即,不满足 $DirREACH_{\varepsilon,\mu}(v,u)$.

**定义 6(结构可达)**. 如果存在一条节点链 $v_1v_2v_3\ldots v_{n-1}$ 中的点都是核心节点,其中 $v_1=v,v_n=u$,若满足 $v_1v_2v_3\ldots v_{n-1}$ 是核心节点,$v_i$ 和 $v_{i+1}$ 之间直接结构可达,即 $DirREACH_{\varepsilon,\mu}(v_i,u_{i+1})$,则就认为 $v$ 和 $u$ 之间是结构可达的,表示为

$$REACH(v,u) \tag{5}$$

**定义 7(结构聚类 cluster)**. 已知 $u\in C$,且 $u$ 是核心节点,对于$\forall u$,若 $u$ 结构可达 $v$,即 $REACH(v,u)$,则 $v\in C$,称为点最大性.对于$\forall v_1,v_2\in C$,有$\forall u\in V$,使得 $REACH(u,v_1)$和 $REACH(u,v_2)$,称为连通性.一个结构聚类需要同时满足点最大性和连通性.

## 2 算法整体研究框架

基于 MapReduce 的结构聚类,其研究框架如图 1 所示.算法要求整体基于 MapReduce 实现,因此在图 1 中由 7 个小部分组成,每部分都是基于 MapReduce 实现的.

图 1 中,算法主要分为分布式计算相似性、维度扩展[9]和分布式合并聚类这 3 个主要部分.

其中,计算相似性分为 4 部分,目的是为了检测核心与非核心节点,主要包括数据处理、顶点度计算、边上三角形个数计算和相似度计算;维度扩展是为了让图的每个节点带上核心节点信息,在后期聚类阶段能够被识别,从而正确聚类;最后是本文提出的两个基于 MapReduce 的结构聚类算法.

具体算法过程在第 3 节详细给出.

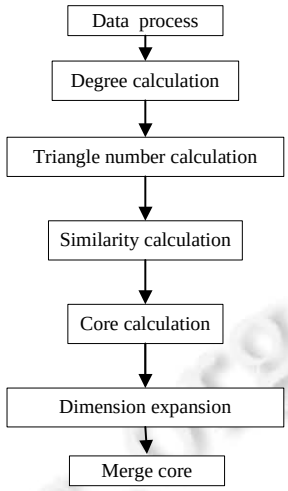



Fig.1 Overall process of distributed SCAN based on MapReduce

图 1 基于 MapReduce 的分布式结构聚类整体流程

## 3 基于 MapReduce 的结构聚类过程

MapReduce 框架很适合处理大规模的流数据,而图算法的实现一直是 MapReduce 的难点.在本节中,详细介绍如何在 MapReduce 上一步步求解出最终用于聚类的核心节点及其直接可达邻居集合,并且对该核心节点以及局部聚类进行合并.

### 3.1 基于MapReduce求图中核心节点

首先,在求结构聚类的时候,需要先求出网络图中的核心节点及其直接可达邻居集合.如图 2 所示,假设当 $\varepsilon$=0 时,图 2(b)是图 2(a)中核心节点 7 与 5 个$\varepsilon$邻居的示例(假设$\varepsilon$满足条件),类似于图 2(b),每个节点都有一个同样性质的子图,这些子图构成的集合是聚类[10]的基础.

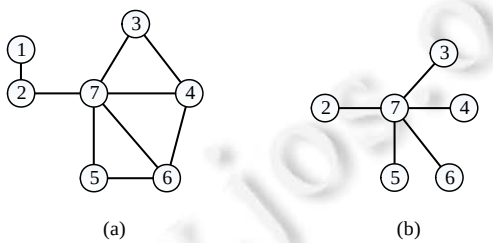


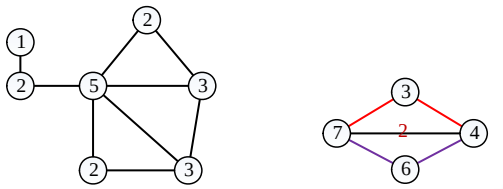

Fig.2 The Original graph and one of the local clusters constituted by cores

图 2 原始无向图及其中某个核心节点构成的局部聚类

计算核心节点分为以下几个部分:节点的度、节点之间的共同邻居(即边的三角形个数)、节点之间的相似性,从而求出核心节点.

首先是求节点的度和边所在的三角形个数,度是图节点最基本的特征,而计算边所在三角形的个数在图结构中具有重要的特征.通过求出边上三角形的个数,可以直接判断求出节点之间的共同邻居个数.在 Hadoop 中,求节点度的方法比较简单,比较有技巧的是求三角形,这两个 MapReduce 算法在其他文献[11]中已经详细介绍.

以图 2(a)为例:节点度和边三角形个数的结果图如图 3 所示,图 3(a)表示顶点的度,图 3(b)表示边(4,7)有两个三角形,分别是Δ347 和Δ467,表示节点 4 和节点 7 有两个共同邻居节点 3 和节点 6.详细的 MapReduce 过程在此不做具体介绍.




(a) 图节点的度 (b) (4,7)边的三角形个数

Fig.3 Node's degree and the edge (4,7)'s triangle number

图 3 无向图节点度和边(4,7)三角形个数

基于带度的边以及边上三角形的个数求图中两点之间的相似性.过程如下.

Map 函数的输入为⟨*key*,*value*⟩对,该键值对有两种:一种是顶点以及顶点度,以⟨*u*|*degree*(*u*) *v*|*degree*(*v*)⟩的形式给出;另一种是边以及边上三角形的个数,以⟨*uv triangleNumber*⟩的形式给出,在算法 1 中,将其当作已知条件作为输入.Map 函数的 *key* 值输出是以边(*u*,*v*)为新的 *key*′,*value* 输出是顶点度 *degree*(*u*)|*degree*(*v*)和三角形个数 *triangleNumber* 作为 *value*′.Reduce 函数的输入是上一轮 Map 函数的输出,输出是边以及边相似性.

**算法 1**. 计算节点之间相似性.

输入:边及其顶点度⟨$edge_{uv}$ {*u*|*degree*(*u*),*v*|*degree*(*v*)}⟩,边及其边上三角形个数⟨$edge_{uv}$ $triangleNumber_{uv}$⟩;

输出:边及其边上的相似性⟨$edge_{uv}$ $similarity_{uv}$⟩.

1. ***Map***(*key*,*value*)
2. *key*′←$edge_{uv}$
3. **if** *value* contains *degree* **then**
4. *value*′←*d*(*u*)|*d*(*v*)
5. **else**
6. *value*′←$similariy_{uv}$
7. *Emit*⟨*key*′,*value*′⟩*pairs*
8. ***Reduce***(*key*′,*values*′)
9. $d_u$←0
10. $d_v$←0
11. $commonNeighbors_{uv}$←0
12. *similarity*←0
13. **for** *value* in *values*′ **do**
14. **if** *value* contains *degree* **then**
15. $d_u$←*degree*(*u*)
16. $d_v$←*degree*(*v*)
17. **else**
18. $commonNeighbors_{uv}$←$triangleNumber_{uv}$
19. $similarity_{uv} = \dfrac{commonNeighbors_{uv} + 2}{\sqrt{(d_u + 1) \times (d_v + 1)}}$
20. *Emit*⟨$edge_{uv}$ $similarity_{uv}$⟩*pairs*

以图 2(a)为例:相似性的结果如图 4 所示,图 4(a)顶点数据表示顶点的度,边上数据表示边的三角形个数;图 4(b)的顶点数据代表节点编号,边上数据是边两端点的相似性.图 5 是详细的以无向图 2(a)为例子求出每条边的相似性的 MapReduce 过程.可以看到:在 Map 阶段,将有序边作为 *key* 值,因此,同一条边的两个顶点的度和共同邻居就会进入到一个 Reduce 中,在 Reduce 中,可以利用这两个信息进行边相似性的计算.

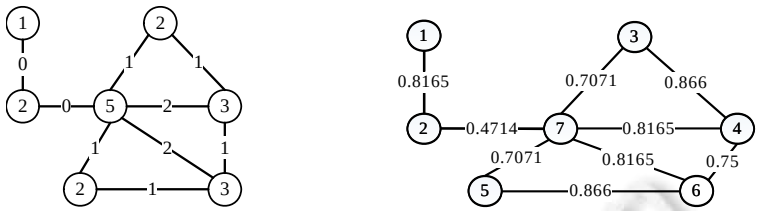



Fig.4 Calculate edge similarity based on the degree and the number of triangles

图 4 基于节点度和边三角形个数计算节点相似性

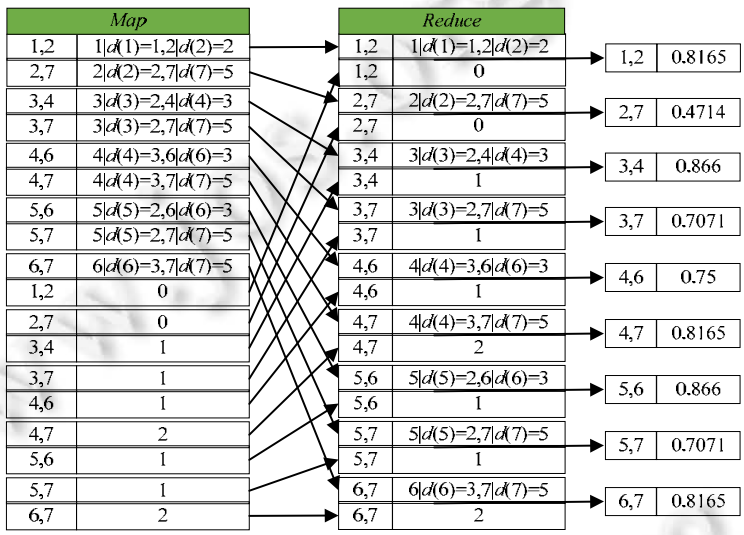



Fig.5 MapReduce process for calculating edge similarity

图 5 计算节点相似性的 MapReduce 过程

接着,本文需要根据边的相似性求出图中的核心节点.由核心节点的定义可知:当节点 $u$ 的 $\varepsilon$ 邻居不小于给定的阈值 $\mu$ 时,$u$ 节点就被定义为核心节点.因此可以设计出检测核心节点的 MapReduce 程序,算法是以边上两点作为 *key*,以边的相似性作为 *value* 构成⟨*u*,*v* *similarity*⟩对输入;以图的每个点作为 *key′*,以布尔类型的变量 *isCore* 来标记该点是否为核心节点作为 *value′*构成⟨*u*,*isCore*⟩对输出,*T* 表示该点是核心,*F* 表示该点非核心.

**算法 2**. 统计核心节点.

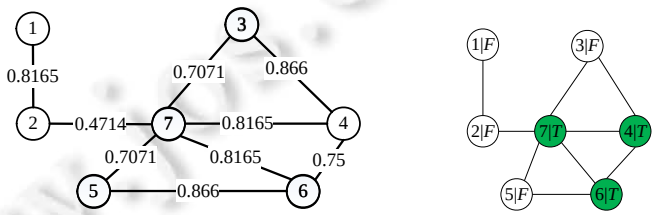
输入:边及其相似性⟨$edge_{uv}similarity_{uv}$⟩,给定 $\varepsilon$,$\mu$;

输出:顶点及其核心变量 *isCore* 的(*u*,*isCore*)对.

1. ***Map***(*key*,*value*)
2. *u*←*key.u*
3. *v*←*key.v*
4. *Emit*⟨*u v*,$similarity_{uv}$⟩*pairs*
5. *Emit*⟨*v u*,$similarity_{uv}$⟩*pairs*
6. ***Reduce***(*key′*,*values′*)
7. *ε-count*←0
8. $d_v$←0
9. **for** *value* **in** *values′* **do**

10. **if** $similarity_{uv} \geqslant \varepsilon$ **then**
11. $\varepsilon$-$count \leftarrow \varepsilon$-$count+1$
12. **if** $\varepsilon$-$count \geqslant \mu$ **then**
13. $value'' \leftarrow T$
14. **else**
15. $value'' \leftarrow F$
16. $Emit\langle u,value''\rangle pairs$

例如,在图 6(a)中,当$\varepsilon$=0.7,$\mu$=3 时,无向图的核心节点统计如图 6(b)所示,核心节点为 4b6 和 7,节点填充颜色的是带 *T* 的核心节点,无颜色的是带 *F* 的非核心节点.详细统计核心节点的 MapReduce 过程如图 7 所示:在 Map 阶段,算法将边及其相似性信息分割成该边的两个节点带上该边的相似性;在 Reduce 阶段,以每个节点作为 *key*,它的邻居和两者的相似性会进入到同一个 *Reduce*.因此,可以在 *Reduce* 阶段判断每个节点有多少个$\varepsilon$邻居,从而判断该点是否为核心节点.




(a) (b)

Fig.6 Core(4,6,7) and non-core(1,2,3,5)

图 6 核心(4,6,7)与非核心节点(1,2,3,5)

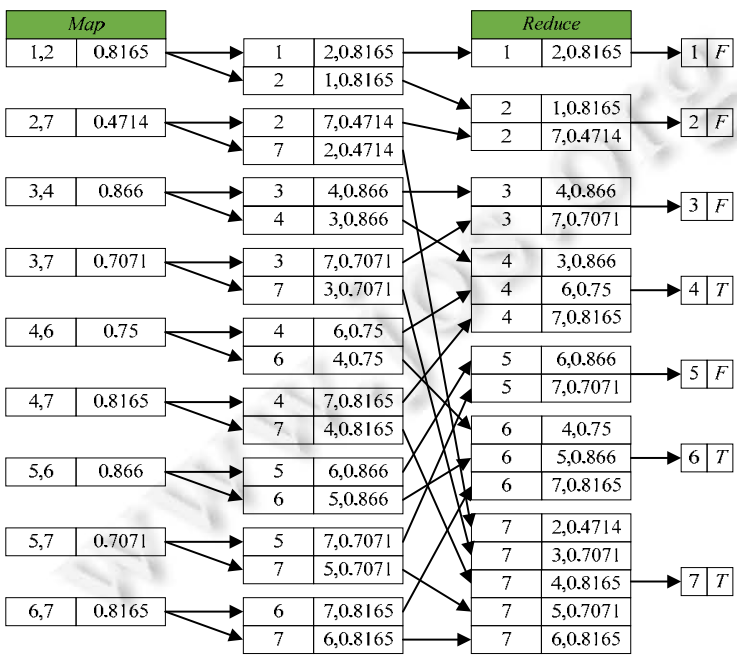



Fig.7 MapReduce process for calculating cores

图 7 计算图中核心节点的 MapReduce 过程

### 3.2 基于MapReduce的图维度扩展

在第 3.1 节中,我们对成功地对图的每个节点进行了是否为核心的标记,但是每个节点还只是带着编号信息与单个节点是否为核心,而且核心节点的所有邻居里面还有一些与核心节点只是邻居,没有满足是核心节点$\varepsilon$邻居的条件.如图 8 所示,图 8(a)是在某个条件得到的核心节点状态,图 8(b)是维度扩展后的没有利用核心和$\varepsilon$剪枝的局部聚类.

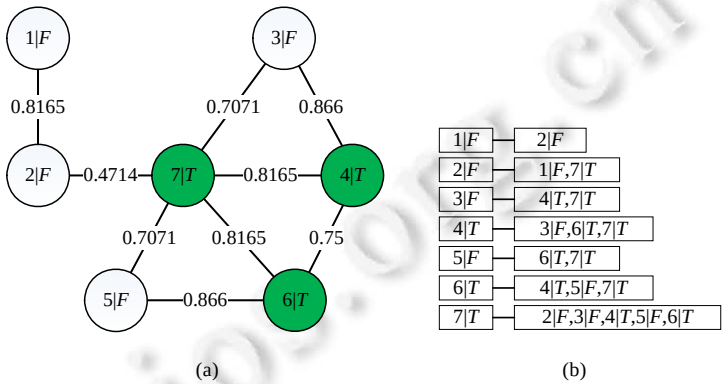



Fig.8 Node dimension extension

图 8 图节点维度扩展

接下来,需将维度扩展后的最初状态剪枝掉与核心不构成$\varepsilon$邻居的节点,并利用核心节点剪枝掉不需要合并的分支.

如图 9 所示,图 9(a)是最初聚类状态,利用核心节点可以去掉 1、2、3、5 这 4 个非核心节点所在的局部聚类,经过剪枝后得到图 9(b).可以看出:最终进入合并的局部聚类数量经过核心节点剪枝后与原来相比大量减少,进入合并的聚类经过$\varepsilon$剪枝也会变少(可以看出,节点 2 不是节点 7 的$\varepsilon$邻居),两者可以同时优化算法的时间和空间 I/O 次数.

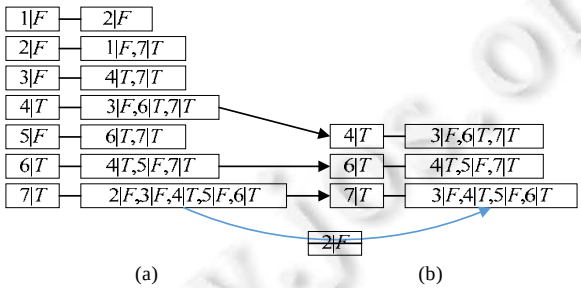



Fig.9 Core-Based and $\varepsilon$-based pruning

图 9 基于核心与$\varepsilon$的剪枝

**算法 3**. 基于核心节点与边相似性的维度扩展与剪枝.

输入:顶点以及核心标记⟨u *isCore*⟩,边的两顶点以及边的相似性⟨*u*,*v* $similarity_{uv}$⟩;

输出:顶点及其直接可达邻居集合构成的键值对⟨*u*, *set*($Neighbor_u$|*isCore*)⟩对.

1. ***Map*1**(*key*,*value*)
2. **if** *value* is *T*||*value* is *F* **then**
3. *Emit*⟨*u*,*isCore*⟩*pairs*
4. **else**
5. **if** $similarity_{uv} \geqslant \varepsilon$

```
6.          Emit⟨u,v⟩pairs
7.   Reduce1(key′,values′)
8.   for value in values′ do
9.          if value is T||value is F then
10.            isCore←value
11.         else
12.            valueList.add(value)
13. value″←key′|isCore
14. for val in valueList do
15.         if key′<val do
16.            key″←key′|val
17.         else
18.            key″←val|key
19. Emit⟨key″,value″⟩pairs
20. Map2(key,value)
21. Emit⟨key′,value′⟩pairs
22. Reduce2(key′,values′)
23. for value in values′ do
24.         valueList.add(value)
25. Emit⟨valueList[1],valueList[0]⟩pairs
26. Emit⟨valueList[0],valueList[1]⟩pairs
27. Map3(key,value)
28. Emit⟨key,value⟩pairs
29. Reduce3(key′,values′)
30. for value in values′ do
31.         value″←value″+value+,
32. value″←value″.length()−1
33. Emit⟨key,value″⟩pairs
```

算法 3 包含了 3 个 MapReduce 程序,MapReduce 过程如下.Map 函数的输入为⟨*key*,*value*⟩键值对,该输入键值对有两种.其中一种是顶点以及 *isCore* 变量,以⟨*u isCore*⟩的形式给出;另一种是边以及边的相似性,以⟨*uv similarity*$_{uv}$⟩的形式给出.

本文基于 MapReduce 的结构聚类的主要创新在于设计了分布式计算相似性以及两种合并聚类的算法,而维度扩展作为处理工具,使得图中每个节点都附带核心信息,在合并聚类中起到至关重要的作用,使得聚类过程能够直接查看到当前节点是否为核心节点,减少了冗杂的判断过程.其 MapReduce 过程如图 10 所示,通过图例中 *Reduce*3 的结果可以直观地看出:经过维度扩展和剪枝[12]后,可以得到每个带有核心信息的节点及其结构可达的邻居.

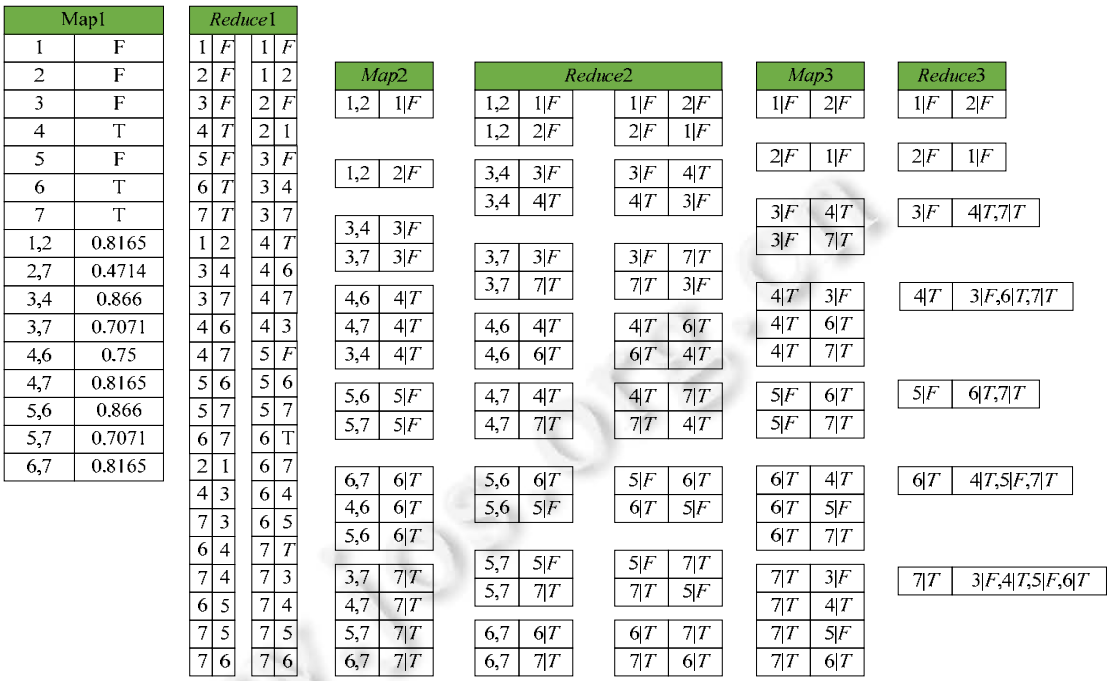



Fig.10 Dimension extension in MapReduce

图 10 维度扩展的 MapReduce 过程

### 3.3 基于MapReduce对分支进行结构聚类

由算法 3,我们可以得到每个节点及其全部$\varepsilon$邻居[13],图中每个节点都是一个元组,以($u$|*isCore*)的形式存在.我们可以通过对图中核心节点及其直接可达邻居进行不断的迭代合并[14],最终得到聚类.迭代合并的停止条件是当前聚类不能加入新的元素,即不能够再继续与其他聚类合并.

过程见算法 4:Map 阶段在可达点中(初始集合里是直接可达邻居)找出是核心的节点,将可达邻居中所有核心 $v$ 作为 *key*′,将当前可达邻居构成的集合 $C_u$ 作为 *value*′输出.在 Reduce 阶段,对相同节点的不同聚类进行合并,迭代得到最终聚类.

**算法 4**. General Union.

输入:维度扩展后的顶点 $u$ 及其直接可达邻居的初级聚类 $C_u$;

输出:聚类.

1. ***Map***($u$,$C_u$)
2. **while** $v\in C_u$ **do**
3. **if** $v$.*isCoreisTrue* **then**
4. *Emit*⟨$v$,$C_u$⟩*pairs*
5. ***Reduce***($u$,{$C_1$….$C_n$})
6. $C\leftarrow$*Merge*($C_1$….$C_n$)
7. *Emit*⟨$u_{\min}$,$C$⟩

MapReduce 聚类原理如图 11 所示,本文给出聚类的核心,算法过程需要用到两个 MapReduc[15],但第 2 个 MapReduce 只是简单的去重[16]过程,具体不给出.

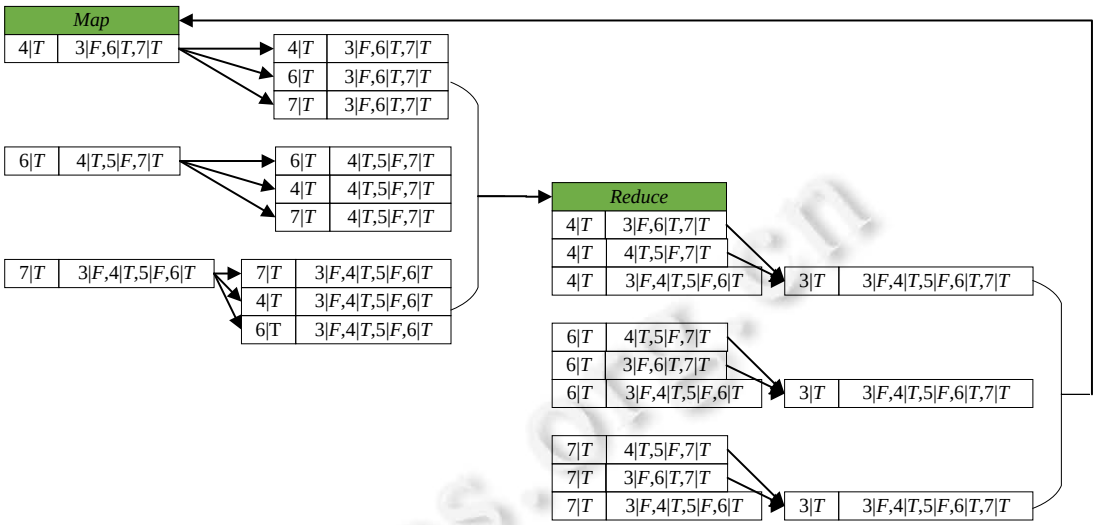



Fig.11 The process of general union

图 11 General union 过程

算法 4 是一般的局部聚类合并[17]算法,同时作为算法 5 的基准实验,在实验结果图中也可以看出两种算法的性能差异.观察算法 4 可以发现:Map 阶段每次都输出了核心节点及当前聚类的所有元素,导致了 IO 次数[18]快速增长,可以考虑使用最小编号的核心节点作为当前局部聚类的标记,从而对算法 4 进行优化.具体如下.

首先,Map 阶段,我们在局部聚类中找出是核心的节点,并且标注最小的核心,将所有核心分别作为 *key*,最小核心作为 *value* 构造成键值对输出.同时,以最小核心为 *key*,将当前局部聚类集合 $C_u$ 作为 *value* 构造成键值对输出,MapReduce 框架会将同一个 *key* 的键值对分到同一个 reduce;然后,在 Reduce 阶段对相同节点的不同聚类进行合并;最后,查看迭代条件,是否需要迭代,得到最终聚类.

基于上述步骤可以看出:在两个分别以 $u,v$ 标记的局部聚类里,假如局部聚类 $C_u,C_v$ 可以合并,$C_u$ 中必存在一点 $v_{\min},v_{\min}$ 也在 $C_v$ 中,利用 $v_{\min}$ 可以将两个聚类联系在一起,合并过程见算法 5,图示过程如图 12 所示,迭代条件与算法 4 相同.

**算法 5**. Base Min_Core Union.

输入:维度扩展后的顶点 $u$ 及其直接可达邻居的初级聚类 $C_u$,已知 $Neighbor[u]$;

输出:聚类.

1. ***Map***$(u,C_u)$
2. $v_{\min}\leftarrow v|v\in C_u\wedge v.isCoreisTrue$
3. **while** $v\in Neighbor[u]$ **do**
4. **if** $v.isCoreisTrue$ **then**
5. $Emit\langle v,v_{\min}\rangle pairs$
6. $Emit\langle v_{\min},C_u\rangle pairs$
7. ***Reduce***$(u,\{C_1....C_n\})$
8. $C\leftarrow Merge(C_1....C_n)$
9. $Emit\langle u,C\rangle$

算法 4、算法 5 中的 $Merge(C_1...C_n)$函数是合并可以合并的两个局部聚类函数,本质上是一个集合的并[19]操作,集合元素类型是点及其核心属性构成的元组.算法 5 中,$Neighbor[u]$数组表示 $u$ 的可达邻居节点.合并函数的具体过程见算法 6.

**算法 6**. Merge SubCluster.

输入:维度扩展后的顶点 $u$ 及其直接可达邻居的初级聚类 $C_u$,已知 $Neighbor[u]$;

输出:聚类 $C$.

1. ***Merge***($C_1$….$C_n$)
2. **for** $i$=0→$n$ **do**
3. **for** $c$ **in** $C_i$
4. **if** $C$ contains $c$ **do**
5. continue
6. **else**
7. $C.add$(c)
8. return $C$

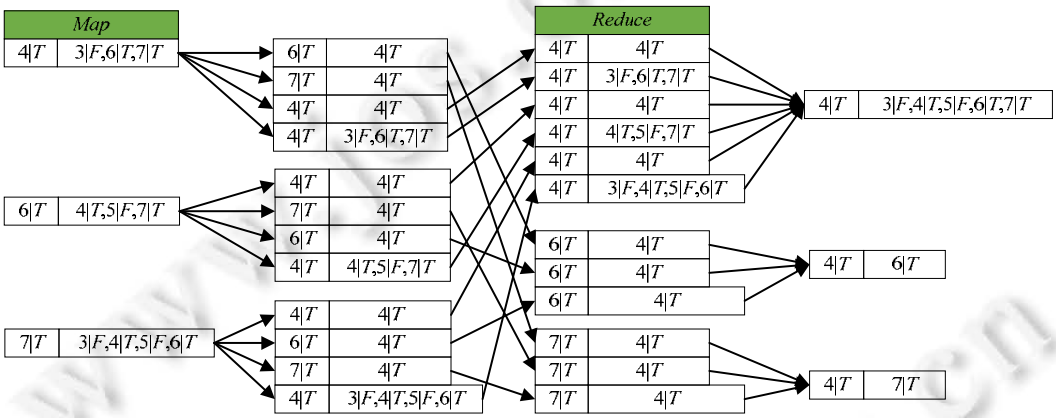



Fig.12 Base min_coremerge subclusters

图 12 最小核局部聚类

## 4 实验与结果

在已搭建的 Hadoop 分布式集群上,使用本文提出的算法构建了一种面向无向图[20]的结构聚类模型,并且在 4 个真实的数据集上(Skitter,Pokec,LiveJournal 和 Com-Orkut)上测试本文提出的方法.

### 4.1 实验平台与实验数据集

实验环境为 8 台服务器构成的 Hadoop 集群,1 个 master,7 个 slaver,每台服务器有两个 CPU,CPU 配置参数为:Intel(R) Xeon(R) CPU E5-2620 v3@2.40GHz,每个 CPU 有 6 个核,共 12 个核,因此,集群一共有 96 个核.实验过程中,我们可以根据集群核数调整 Reduce 个数.单台服务器有 32GB 的内存,统一安装 Linux ubuntu16.04 64 位操作系统.

实验数据集来源于斯坦福大学的公开数据集.数据的详细信息见表 1,从 4 个数据集的边数可以看出,实验数据具有一定的梯度性[21].

**Table 1** Experimental data set and its vertex number and edge number

**表 1** 实验数据集及其顶点数和边数

| DataSet | Number of nodes | Number of edges |
|---|---|---|
| Skitter | 1 696 415 | 11 095 298 |
| Pokec | 1 632 000 | 22 301 000 |
| LiveJournal | 3 997 962 | 34 681 189 |
| Com-Orkut | 3 072 441 | 117 185 083 |

### 4.2 实验结果及分析

实验采用 4 个数据集,相似性$\varepsilon$和$\varepsilon$邻居数$\mu$这两个参数分别取(0.4,3)和(0.6,6),实验收集得到 8 个实验结果图如图 13 所示.

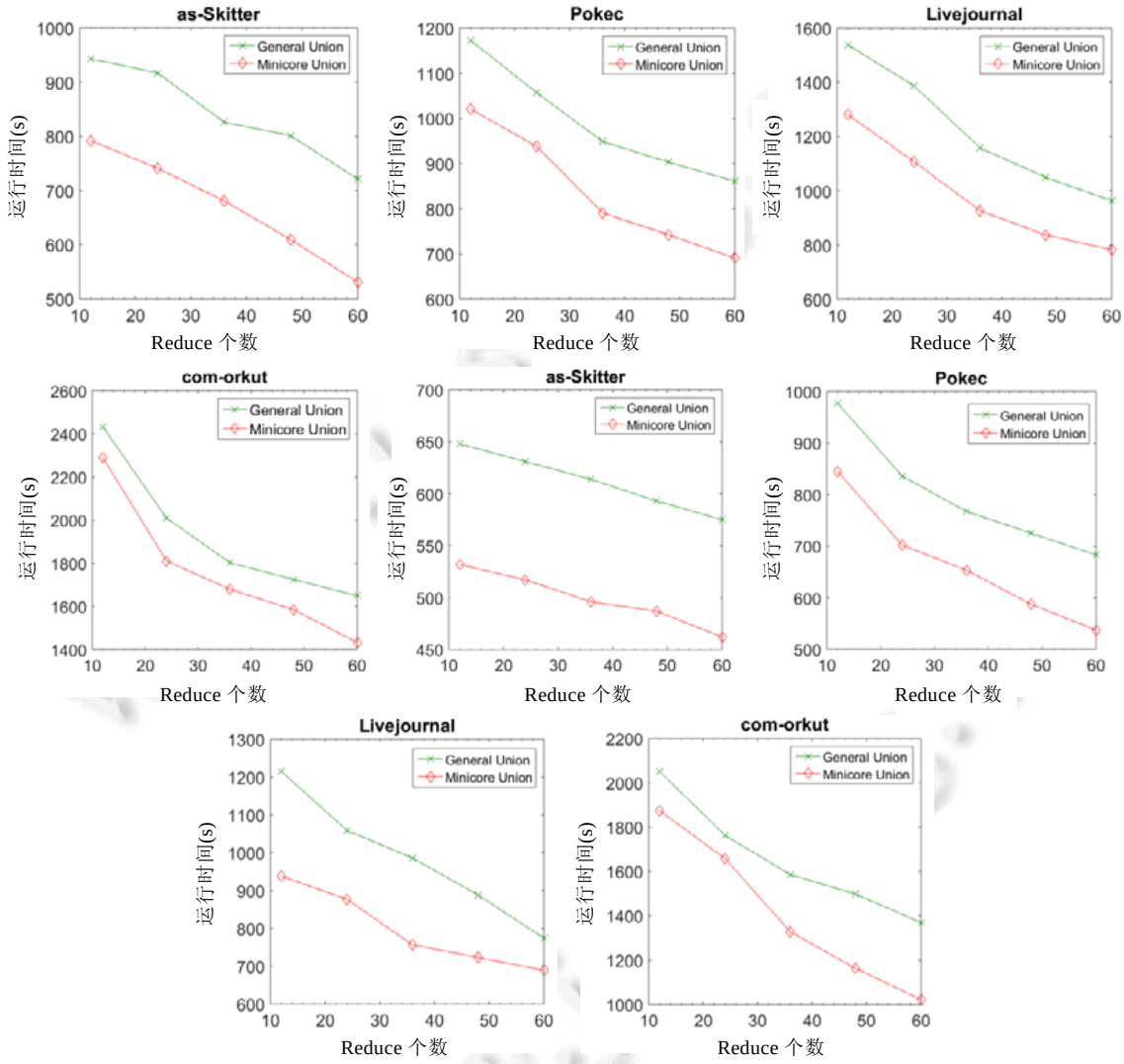



Fig.13 Efficiency comparison varying number of reduce

图 13 不同 reduce 数量运行效率对比

在图 13 的实验结果中:前 4 个结果图采用了统一的$\varepsilon$和$\mu$,取值(0.6,6),后 4 个结果图取值(0.4,3),并且让 reduce 个数线性增加(12,24,36,48,60)来观察实验结果.这里不需要设置 *Map* 个数,*Map* 个数是 Hadoop 集群根据数据集大小自动进行分配.

- 观察 1.单独观察图 13 每个实验结果可以发现:当数据集不变时,reduce 个数增加,算法的运行时间大致呈线性降低;
- 观察 2.根据实验结果可以发现,算法 5(Base Min_Core Union)比算法 4(General Union)的合并效率高;
- 观察 3.当 reduce 个数一样时,数据集的边数越大,算法时间越长;

- 观察 4.$\varepsilon$和$\mu$的取值越大,剪枝效率越高.

## 5 总 结

为了解决图结构聚类算法的可扩展性问题,本文提出一种基于 MapReduce 的结构聚类算法 MRSCAN.具体地,本文设计了一套计算核心节点以及两种有效的合并聚类的 MapReduce 算法.最后,在多个真实的大规模图数据集上进行测试,实验结果验证了本文算法的正确性、有效性以及可扩展能力.未来的工作包括:将我们实现的算法推广至其他的分布式计算框架,例如 Pregel[22]计算框架.

**References**:

[1] Mayer-SchoSnberger V, Cukier K, Wrote; Zhou T, *et al*. Trans. Big Data: A Revolution That Will Transform How We Live, Work, and Think. John Murray, 2013 (in Chinese).

[2] Shvachko K, Kuang H, Radia S, Chansler R. The hadoop distributed file system. In: Proc. of the IEEE Symp. on MASS Storage Systems and Technologies. IEEE Computer Society, 2010. 1–10. [doi: 10.1109/MSST.2010.5496972]

[3] Shiokawa H, Fujiwara Y, Onizuka M. SCAN++: Efficient algorithm for finding clusters, hubs and outliers on large-scale graphs. Proc. of the VLDB Endowment, 2015. [doi: 10.14778/2809974.2809980]

[4] Chang L, Li W, Qin L, Zhang W. pSCAN: Fast and exact structural graph clustering. IEEE Trans. on Knowledge & Data Engineering, 2017,29(2):387−401. [doi: 10.1109/TKDE.2016.2618795]

[5] Li JJ, Cui J, Wang D, Yan L, Huang YS. Survey of MapReduce parallel programming model. Acta Electronic Journal, 2011,39(11):2635−2642 (in Chinese with English abstract).

[6] Wang F, Lei BH. Model analysis of hadoop distributed file system. Telecommunications Science, 2010,26(12):95–99 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-0801.2010.12.019]

[7] Chen F, Kodialam M, Lakshman TV. Joint scheduling of processing and Shuffle phases in MapReduce systems. In: Proc. of the IEEE INFOCOM. IEEE, 2012. 1143−1151. [doi: 10.1109/INFCOM.2012.6195473]

[8] Xu X, Yuruk N, Feng Z, Schweiger TAJ. SCAN: A structural clustering algorithm for networks. In: Proc. of the ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2007. 824–833. [doi: 10.1145/1281192.1281280]

[9] Zhou FF, Li JC, Huang W, Wang JW, Zhao Y. Based on dimension expansion Radviz visual clustering analysis method. Ruan Jian Xue Bao/ Journal of Software, 2016,27(5):1127−1139 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4951.htm [doi: 10.13328/j.cnki.jos.004951]

[10] Guo QK. Study on connection method based on MapReduce [Ph.D. Thesis]. Jilin University, 2014 (in Chinese with English abstract).

[11] Cohen J. Graph twiddling in a MapReduce world. Computing in Science & Engineering, 2009,11(4):29–41. [doi: 10.1109/MCSE.2009.120]

[12] Wang HY, Zhou L. Study on the maximum mission problem based on divide, prune and ant colony algorithm. Journal of Hefei Teachers College, 2011,29(3):59−62 (in Chinese with English abstract). [doi: 10.3969/j.issn.1674-2273.2011.03.020]

[13] Qin L, Yu J X, Chang L, Cheng H, Zhang C, Lin XM. Scalable big graph processing in MapReduce. In: Proc. of the SIGMOD. 2014. 827−838. [doi: 10.1145/2588555.2593661]

[14] Feng XJ. Research on iterative distributed data processing based on MapReduce [Ph.D. Thesis]. Shandong University, 2013 (in Chinese with English abstract).

[15] Liu T, Wu SH, Qiang Y. Task scheduling algorithm for multiple MapReduce jobs. Microelectronics & Computer, 2013,30(12):156−159 (in Chinese with English abstract).

[16] Gao SL. Research on Web page parallel de-emphasis algorithm based on MapReduce framework. Heilongjiang Science, 2010,1(5):13−18 (in Chinese with English abstract).

[17] Que X, Wang Y, Xu C, Yu WK. Hierarchical merge for scalable MapReduce. In: Proc. of the Workshop on Management of Big Data Systems. 2012. 1−6. [doi: 10.1145/2378356.2378358]

[18] Tiwari N, Sarkar S, Indrawan-Santiago M, Bellur U. Improving energy efficiency of IO-intensive MapReduce jobs. In: Proc. of the Int'l Conf. 2015. 1−4. [doi: 10.1145/2684464.2684484]

[19] Zhang X. Design of DBSCAN algorithm based on query and search. Journal of Yili Normal University: Natural Science Edition, 2014,8(4):62−65 (in Chinese with English abstract). [doi: 10.3969/j.issn.1673-999X.2014.04.014]

[20] Ma J, Iwama K, Ma SH. Search for parallel algorithm in loop of undirected graph. Ruan Jian Xue Bao/Journal of Software, 1997, 18(6):475−480 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/475.htm

[21] Hou ZL, Wei XH, Huang DN, Xu S. Application of parallel computing and its performance analysis in gravity totally gradient data inversion. Applied Geophysics, 2015,12(3):292−302 (in Chinese with English abstract).

[22] Malewicz G, Austern MH, Bik AJC, Dehnert JC, Horn I, Leiser N, Czajkowski G. Pregel: A system for large-scale graph processing. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. ACM Press, 2010. 135−146. [doi: 10.1145/1582716.1582723]

**附中文参考文献:**

[1] Mayer-Schonborger V, Cukier K,著;周涛,等,译.大数据时代:生活、工作与思维的大变革.杭州:浙江人民出版社,2013.

[5] 李建江,崔健,王聃,严林,黄义双.MapReduce 并行编程模型研究综述.电子学报,2011,39(11):2635−2642.

[6] 王峰,雷葆华.Hadoop 分布式文件系统的模型分析.电信科学,2010,26(12):95−99. [doi: 10.3969/j.issn.1000-0801.2010.12.019]

[9] 周芳芳,李俊材,黄伟,王俊韡,赵颖.基于维度扩展的 Radviz 可视化聚类分析方法.软件学报,2016,27(5):1127−1139. http://www.jos.org.cn/1000-9825/4951.htm

[10] 郭骐恺.基于 MapReduce 的连接方法研究[博士学位论文].长春:吉林大学,2014.

[12] 王会颖,周琳.基于分治、剪枝和蚁群算法求解最大团问题.合肥师范学院学报,2011,29(3):59−62. [doi: 10.3969/j.issn.1674-2273.2011.03.020]

[14] 冯新建.基于 MapReduce 的迭代型分布式数据处理研究[博士学位论文].济南:山东大学,2013.

[15] 刘涛,武淑红,强彦.用于多个 MapReduce 作业的任务调度算法.微电子学与计算机,2013,30(12):156−159.

[16] 高殊丽.基于 MapReduce 框架的网页并行去重算法研究.黑龙江科学,2010,1(5):13−18.

[19] 张晓.基于并查集的 DBSCAN 算法设计.伊犁师范学院学报:自然科学版,2014,8(4):62−65. [doi: 10.3969/j.issn.1673-999X.2014.04.014]

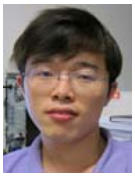
[20] 马军,岩间一雄,马绍汉.寻找无向图中回路的并行算法.软件学报,1997,18(6):475−480. http://www.jos.org.cn/1000-9825/18/475.htm

**张伟鹏**(1991－),男,广东汕头人,硕士生,主要研究领域为大规模图数据并行算法.

**刘宇鸿**(1997－),男,学士,主要研究领域为社区搜索,时态图挖掘.

**李振军**(1979－),男,博士,工程师,主要研究领域为数据挖掘,深度学习.

**毛睿**(1975－),男,博士,教授,CCF 高级会员,主要研究领域为数据挖掘,数据库,统计方法,机器学习,计算生物.



**李荣华**(1985－),男,博士,讲师,主要研究领域为图数据挖掘,社交网络分析.

**乔少杰**(1981－),男,博士,教授,CCF 高级会员,主要研究领域为轨迹数据挖掘,机器学习.