TSCAN:利用并行策略改进的图结构聚类算法

陈亚中[1],李振军[1],李荣华[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,成都 610103

摘 要:近年来,图数据聚类在学术界引起了广泛的关注,许多优秀的聚类方法,如模块度优化算法、谱聚类,以及基于密度的聚类算法在图数据上取得了很好的效果。SCAN是一种著名的基于密度的图聚类算法,该算法不仅能够找出图中的聚类,而且还能够发现不同聚类间的Hub节点,以及图中的离群点。然而,该算法存在两方面的局限性:首先,在大规模图数据上,该算法需要耗费大量的时间用于计算图中每条边的结构相似性;另一方面,该算法存在两个参数ε和μ,并且对这两个参数比较敏感。为了解决其局限性,提出了一种基于OpenMP的并行算法来求解节点相似性,并且提出了两种有效的负载均衡策略;其次,提出一种基于三角形的新型图结构聚类算法TSCAN。该模型能够有效降低算法对参数的敏感性,而且还能够发现重叠以及更稠密的社区。在多个大规模数据集上实验发现,基于多核的并行算法能够达到近乎线性的加速比,而且TSCAN算法对参数不敏感,能有效发现重叠社区。

关键词:社区探测;结构聚类算法;重叠社区;OpenMP;并行算法
文献标志码:A **中图分类号:**TP302.8 doi:10.3778/j.issn.1002-8331.1710-0230

陈亚中,李振军,李荣华,等.TSCAN:利用并行策略改进的图结构聚类算法.计算机工程与应用,2019,55(1):76-83.
CHEN Yazhong, LI Zhenjun, LI Ronghua, et al. Improved graph structure clustering algorithm by using parallel strategy. Computer Engineering and Applications, 2019, 55(1):76-83.

Improved Graph Structure Clustering Algorithm by Using Parallel Strategy

CEHN Yazhong[1], LI Zhenjun[1], LI Ronghua[1], MAO Rui[1], QIAO Shaojie[2]
1.College of Computer Science & Software Engineering, Shenzhen University, Shenzhen, Guandong 518060, China
2.Chengdu University of Information Technology, Chengdu 610103, China

Abstract: Recently, graph clustering has been attracted much attention in the research community. There are a number of clustering methods, such as modularity optimization algorithm, spectral clustering algorithm, as well as density-based clustering algorithm, that are proved to be useful for graph data. Among those algorithms, the SCAN algorithm is a well-known density-based algorithm for graph data. SCAN is not only able to find clusters, but it also identifies hub nodes and outliers. The SCAN algorithm, however, has two limitations. Firstly, it is very costly to compute the structural similarity for each edge in a massive graph. Secondly, SCAN is sensitive to its parameters ε and μ. To overcome these two limitations, this paper proposes an OpenMP-based parallel algorithm with two carefully-designed load-balancing techniques to compute the similarities efficiently, and a novel triangle-based graph structural clustering algorithm, called TSCAN is proposed. The striking feature of the TSCAN algorithm is that it is not sensitive to the parameters, and it is also capable of finding overlapping and dense communities. Finally, this paper conducts extensive experiments over several real-world datasets to evaluate the proposed algorithms. The results indicate that parallel implementation can achieve near-linear speedup, and the TSCAN algorithm is robust to its parameters and can identify overlapping communities.

Key words: community detection; SCAN algorithm; overlapping community; openMP; parallel algorithm

基金项目:国家自然科学青年基金(No.61402292);国家自然科学基金广东省联合基金(No.U1301252)。

作者简介:陈亚中(1990—),男,硕士研究生,主要研究领域为数据挖掘,E-mail:874908722@qq.com;李振军(1979—),通讯作者,男,博士,工程师,研究领域为大数据、人工智能;李荣华(1985—),男,博士,讲师,主要研究图数据挖掘、时态大数据;毛睿(1975—),男,博士,教授,主要从事数据挖掘的研究;乔少杰(1981—),男,博士,教授,研究方向为云计算与人工智能。

收稿日期:2017-10-24 **修回日期:**2017-12-01 **文章编号:**1002-8331(2019)01-0076-08
CNKI**网络出版:**2018-04-09, http://kns.cnki.net/kcms/detail/11.2127.TP.20180409.1455.020.html

1 引言

随着信息技术的快速发展,各行业积累了大量的业务数据,这些数据集内部相互之间存在着一定的联系,使得每个数据集就可以转换成一个大的图。比如社交网络,生物中的各种分子之间的网路以及交通网络等,深入挖掘这些网络中的内在信息具有重要意义,特别是社区探测近年来更是成为研究的热点。例如在社交网络中,一个社区往往代表着相互之间经常交互的一个团体;而在生物网络中一个社区可能是分子结构相似、性质相同的物质。社区探测能带来巨大的商业价值,吸引众多学者去研究分析,因而产生了众多优秀的图聚类算法和模型。例如,查找图中所有的团以及极大团算法、基于顶点度提出的K-core[1]模型、基于三角形的K-truss[2-3]模型、K-edge[4]模型,此外还有基于密度的聚类模型OPTICS[5]以及基于马尔可夫的聚类算法[6]。其中,基于结构的聚类算法SCAN[7](Structural Clustering Algorithm for Networks)在处理分析现实数据形成的大图时,取得了很好的效果。该算法在发现社区的同时,还能够检测到Hub节点(该点不属于任何聚类,但是连接着不同的聚类,如同在不同的聚类之间架起了一座桥梁),以及离群节点(一些不属于任何聚类的非Hub节点)。鉴于这些优势,在本文中,重点研究SCAN算法。

SCAN在聚类的过程中需要指定两个阈值:判断有边顶点之间是否满足结构相似性的阈值(ε)以及判断顶点是否为核心顶点的阈值(μ)。在SCAN算法中,需要对每条边计算其两个端点之间的结构相似性。当数据过于庞大时,计算整个图中顶点的结构相似性极为耗时,其复杂度是$O(m1.5)$。另外,SCAN算法对参数ε非常敏感,稍有偏差都会使得聚类结果大不相同,甚至产出非合理聚类的问题。SCAN能够产生部分重叠的社区(社区的边界节点可以重叠),但是它的核心节点不能在重叠的社区中,因此在重叠社区问题上也有一定的局限性。

为了解决SCAN算法的计算速度问题,Shiokawa等人[8]提出了剪枝的SCAN算法SCAN++,Chang等人提出了pSCAN[9]。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。在本文中,提出采用并行的计算方法来计算结构相似性,从而达到加速运算的目的。并行计算是解决庞大计算的有效方法,如何对社区探测的各类模型并行化处理,很多学者进行了深入的研究。例如基于MapReduce[10-11]在大规模图数据中列举所有的三角形[12-14];在MPI[15-17]框架上基于空间密度的聚类[18]以及并行枚举图中所有的团(Clique)[19-20]。然而无论是MPI还是MapReduce均基于多机处理,而不同机器间的数据交换不但会增加时间开销,而且会大大增加编程的复杂性。考虑到以上诸多因素,本文提出基于共享内存的多核计算的方法来计算图结构相似性。在对大规模数据进行处理时,算法优先考虑并行计算而不是单机处理,并在实验中取得了很好的效果。

在本文的算法中,所有的数据都存放在内存中(由于内存技术的飞速发展,普通服务器的内存容量也越做越大,目前绝大多数公开的图数据都可以放在一台服务器的内存中),通过线程访问共享数据,在消除多机之间数据交换开销的同时,降低编程复杂性。同时算法强调对共享数据的访问的安全性,合理使用锁机制,使得各线程之间能够同步工作,避免产生资源的不合理竞争发生计算错误或者死锁。

其次,为了解决SCAN算法对相似性参数的敏感问题,提出了一个基于三角形的图结构聚类算法(Triangle SCAN,TSCAN)。TSCAN算法能够降低传统SCAN算法对相似性阈值ε的敏感性,同时还允许核心节点处于不同的重叠社区中,并且还能够得到连接更加紧密的社区。

本文的主要贡献如下:

(1)为了提高算法运行速度,在计算图的结构相似性时,本文使用了基于共享内存的并行计算框架OpenMP,提出了一个并行算法,使得该过程的计算时间随着CPU核数的增加,线性降低。

(2)提出了一种新型的三角形结构聚类算法TSCAN,降低了算法对参数的敏感性,聚类结果更加稳定。

(3)TSCAN算法是基于包含三角形的边进行扩展,所以可以发现更为紧密的社区,以及处于重叠社区的核心点,更加符合现实情况。

(4)在大规模的真实数据上进行了测试,实验结果验证了此结论。

2 问题的描述和相关定义

基于结构的聚类算法在图数据挖掘和分析中具有重要的意义,尤其在社区探测应用中取得了很好的效果,因此也吸引了大量学者做了相关方面的研究。本文研究的图结构聚类算法包含了以下关键定义。

为了更好地理解基于三角形的结构聚类算法,首先给出SCAN算法的相关定义。假定图为$G=(V,E)$。

定义1(顶点结构相似性) 若顶点u、v之间存在边,则u和v的结构相似性的值为:

$$\sigma(u,v)=\frac{|\Gamma(v)\cap\Gamma(u)|}{\sqrt{|\Gamma(v)||\Gamma(u)|}}$$

其中$\Gamma(v)$和$\Gamma(u)$分别表示v、u的邻居个数包括v、u自身,即如果u的邻居个数为5,则$\Gamma(u)$的值为6。

定义2(ε-邻居) 顶点u的ε邻居即和u之间的结构相似性大于等于阈值ε的顶点的集合:

$$N_\varepsilon(u)=\{w\in\Gamma(u)|\sigma(u,w)\geqslant\varepsilon\}$$

定义3(核心顶点) 如果一个顶点的ε邻居的个数

大于等于阈值 μ，则称之为核心顶点：

$$core_{\varepsilon,\mu}(u) \Leftrightarrow |N_\varepsilon(u)| \geqslant \mu$$

定义4（直接结构可达）如果 u 直接结构可达 v，当且仅当 u 是核心顶点且 u 和 v 之间的结构相似性大于等于 ε：

$$DirREACH_{\varepsilon,\mu}(u,v) \Leftrightarrow core_{\varepsilon,\mu}(u) \wedge v \in N_\varepsilon(u)$$

定义5（结构可达）如果 u 结构可达 v 当且仅当存在 $v_1, v_2, \cdots, v_n$ 并且存在 $v_1 = u$，$v_n = v$ 并且存在 $i \in \{1, 2, \cdots, n-1\}$ 使得 $DirREACH_{\varepsilon,\mu}(v_i, v_{i+1})$。

基于SCAN算法，下面给出TSCAN算法的相关定义。

定义6（三角形邻接）给定图中的两个三角形 $\triangle_1$、$\triangle_2$、$\triangle_1$ 与 $\triangle_2$ 邻接，当且仅当它们之间存在至少一条公共边。

定义7（三角形连通）$\triangle_s$ 与 $\triangle_t$ 连通，当且仅当存在 $\triangle_1, \triangle_2, \cdots, \triangle_n (n \geqslant 2), \triangle_1 = \triangle_s, \triangle_n = \triangle_t$，对于所有的 $1 \leqslant i \leqslant n$，均满足三角形邻接。

定义8（聚类）给定一个图 G，非空子图 G' 是一个三角形结构聚类当且仅当 G' 满足如下三个条件：

（1）G' 中每个顶点之间满足结构可达；

（2）$\forall e_1, e_2 \in G'$ 在 G' 中 $\exists \triangle_1, \triangle_2$ 并且有 $e_1 \in \triangle_1$，$e_2 \in \triangle_2$，使得 $\triangle_1 = \triangle_2$ 或者 $\triangle_1$ 与 $\triangle_2$ 三角形连通；

（3）是满足条件（1）和（2）的最大子图，即 $\nexists G'' \subseteq G$ 使得 $G'' \subset G'$ 并且 G'' 满足条件（1）和（2）。

注意：根据定义，很容易得到聚类子图 G' 中的每条边之间不仅满足对称性而且满足传递性。

对称性：对于任意的 $e_m, e_n \in E'$，如果 e_m 三角形可达 e_n，则 e_n 三角形可达 e_m。

传递性：$e_m, e_n, e_p \in E'$，如果 e_m 可达 e_n，e_n 可达 e_p 则 e_m 可达 e_p。

根据上述定义，以图1为例，当 μ=2，ε=0.4时，算法应该分别得到图中的两个聚类 C_1 和 C_2。顶点3和顶点4在 μ=2，ε=0.4下，虽然结构可达，但是不满足三角形连通性的要求，所以虽然二者都是核心顶点也会被划分到不同的聚类中，这也更加符合现实中的情况。此外在聚类 C_1 以及 C_2 中，发现每个小三角形都满足三角形连通性的要求，即聚类中的每条边至少被一个三角形包围；相邻的三角形之间至少有一条公共边。

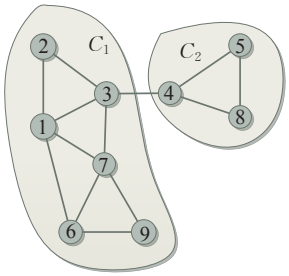


图1　基于TSCAN模型得到的两个聚类

基于定义8，给出此问题描述如下。

问题的描述：给定一个图 G，参数 ε 和 μ，根据定义8求出图中所有的聚类。

3　基于OpenMP计算结构相似性并行策略

本章提出了一种基于OpenMP多核的并行计算方法，用于加快计算节点之间的结构相似性（定义1）。在如何做到负载均衡是设计并行算法的关键之处.如果负载均衡策略设计不当，多核可能比单核的运行速度更慢。即使将资源平均分配给所有的核心，不一定能保证能达到负载均衡的效果[16]。接下来将在上述基础上介绍两种计算结构相似性的并行策略。

3.1　基于边度的负载均衡

根据顶点结构相似性定义（定义1），计算顶点 u、v 之间的相似性，需要找到顶点间的共同邻居，因此需要遍历该顶点 u、v 的邻接链表，该步骤的时间复杂度为 u、v 的顶点度之和 $d(u)+d(v)$。因此，计算所有有边顶点之间的相似性的时间复杂度为：$\sum_{(u,v)\subseteq E} d(u)+d(v)$。为了表述方便，将有边顶点的度之和定义为该边的度：$edge(e_{u,v}) = d(u)+d(v)$。因此，可以首先计算出所有边度的总和：$\sum_{e\subseteq E} edge(e)$；然后根据CPU核心的个数（假设为 k）找出集合 Φ（存储图中边的集合）中所有的 k 等分点，并将所有等份边的集合分别分配给 k 个核心。具体情况如图2所示。从中可以发现，每等份的边的个数是不一定相同的，但是每等份中所有边的度之和（$\sum_{e\subseteq e_k} edge(e)$）是相同的。

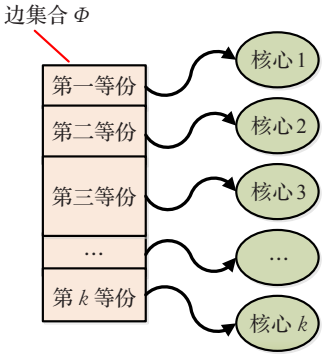


图2　将所有的边按度均分给 k 个核心

算法1的基于边度的相似性计算算法的主要步骤如下所示：

步骤1　计算 Φ 中所有边的度的总和；

步骤2　找出每个 k 等分点在 Φ 的位置；

步骤3　计算所有边的结构相似性 σ，判断定点是否为核心。

算法1　基于度计算结构相似性

输入：Φ 边的集合，k 个核心的个数，ε 核心节点判断阈值

输出：所有边的相似性 σ

```
functionDEGREEBASE(ε,μ)
1.  sum←0
2.  c←1
3.  position[k]←0
4.  #pragma omp parallel for num_threads(k) reduction(+:=sum)
5.  for i←0→Φ.size() do
6.  sum+=Φ[i].edge
7.  计算所有等份的位置并存在 position 数组中
8.  #pragma omp parallel num_threads(k)
9.  {
10. THREAD_ID=opm.get_thead_num()
11. for p=position[THERAD_ID]→
    position[THERAD_ID+1] do
12.   Compute edge similarity σ
13. if σ≥ε then
14. LOCK
15. ε.sum++
16. UNLOCK
17. if ε.sum≥μ then
18. LOCK
19. core_flag=1
20. UNLOCK
21. }
22. return σ
```

算法1的伪代码:在算法1的伪码中,其4至7行是并行计算所有的边度之和的过程,即 $sum=\sum_{e\subseteq\Phi} edge(e)$,随后根据得到的所有边的度总和 sum,计算得到存储仓库 Φ 中所有 k(CPU核心个数)等份点的位置,并将结果保存在数组*position*中;8到22行是并行计算每条边的相似性;根据得到的相似性的结果,同时修改顶点的 ε_num(直接可达邻居个数)字段以及 $core\_flag$(核心节点标签)字段的值。由于这两个字段是所有核心共享进行写访问,所以必须加锁进行同步访问控制,以免发生死锁和同时修改数据情况的发生,造成对结果正确性的影响。

3.2 基于切片的负载均衡

如果将 Φ 中的边平均分配给CPU的 k 个核心去处理,由于 k 值较小(核心个数),而现实网络所形成的图,边数至少上千万条,这样使得每个核分配的边的基数庞大,这极有可能就会造成每个核的完成所得到的计算任务时间差别很大,产生不同核心之间的互相等待,从而影响计算效率。正是基于此,可以利用动态任务分配策略来进行任务的分配调度。即将 Φ 中的所有边以固定的大小分割成 n 个切片(slice),随后采用动态调度机制将每个任务切片循环分配给每个等待计算任务的核心进行计算,只要该核心计算完成就立即向 Φ 申请新的任务。但是,哪个核心处理哪个切片是无法预知的,只要任何一个核心任务结束他都将继续申请新的任务,直到所有的任务都处理完成,这样做的好处是,任务之间无需等待,各自执行,直到最后一个切片被计算完成。整个任务的分配过程如图3所示。在图3中发现,Φ 中的每个切片大小是相同的,即每个切片里面包含的边数一样。算法运行时,核心1到核心 k 分别向边集合 Φ 申请计算任务,系统将前 k 个切片分别分配给 k 个核心的线程进行处理。但是,第 $k+1$ 个任务切片分配给哪个核心是无法事先确定的,只要这 k 个核心中某个线程率先完成已经分配到的任务,它将得到第 $k+1$ 个任务切片。如图3所示,红色虚线表示不确定切片 n 被哪个核心处理,切片 n 可能会被分配给 k 核心中的任何一个首先完成自身任务的核心去处理。

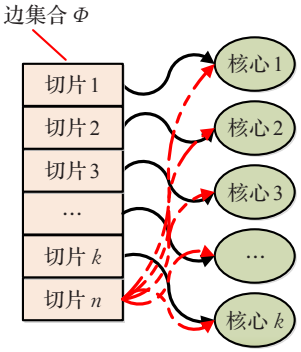


图3 将切片动态分配给每个核心

根据上述分析得到基于切片(slice)的并行计算节点间结构相似性的算法,伪码如算法2所示。

在算法2中,$shared(\varepsilon,\mu)$ 表示阈值参数 ε 和 μ 是所有CPU核心的共享变量,$num\_threads(k)$ 说明算法运行过程中将启动 k 个CPU核心,共同完成所有的计算任务,$schedule(dynamic,\Upsilon)$,表示算法采取的任务分配调度策略方式,其中 $dynamic$ 说明算法采用动态任务调度策略,Υ 表示每份切片(slice)的大小。在实验中,将根据图的大小(图所包含的边数)尝试不同大小的切片来对比算法的运行效率。

算法2 基于切片计算结构相似性

输入:Φ 边的集合,k 核心的个数,μ 核心节点判断阈值,ε 相似性阈值,Υ 切片(slice)大小

输出:所有边的结构相似 σ

```
function SLICEBASE(Φ,k,ε,μ,Υ)
1.  #pragma omp parallel for shared(ε,μ)num_threads(k)
schedule(dynamic,Υ)
2.  for i=0→Φ.size() do
3.  Compute edge similarity σ
4.  If σ≥ε then
5.    LOCK
6.  ε_num++
7.  UNLOCK
```

8. If ε_num≥μ then
9. LOCK
10. core_flag++
11. UNLOCK
12. return σ

4 TSCAN 算法

4.1 三角形结构聚类模型概述

查找三角形在图论领域是一个经典的问题,尤其在社交网络中更是具有重要的意义,因为三角形具有很多优越的性质,比如可以利用它们来推理潜在的社会进程;此外,三角形其自身就是一个团,具有相当稳定的结构。正是由于三角形的诸多优点,在网络中检测三角形吸引了众多学者进行相关研究,并且提出了很多算法;三角形模型也在其他社区探测中作为重要的子过程被广泛应用[2-3]。

传统的结构聚类算法对于参数 ε 以及 μ 极为敏感,然而这两个参数均需人为设定(根据经验指定),尤其是两顶点之间的结构相似性参数 ε,稍有不同,将会形成完全不同的聚类。例如在图 1 中,当设置 μ=2, ε=0.4 时,根据传统的结构聚类算法将得到如图 4 所示的聚类结果。

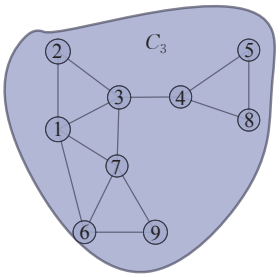


图 4　基于传统模型得到的聚类

然而,当设置 μ=2, ε=0.5 时,可以得到如图 1 所示的聚类 C_1 和 C_2;当参数 ε 设置为 0.4 时,顶点 3 以及顶点 4 都为核心顶点,且互相直接可达,因此可以合并图 1 中的聚类 C_1 和 C_2,从而得到聚类 C_3。为了降低聚类结果对参数 ε 的敏感性,本文率先提出了基于三角形的结构聚类模型,同时在基于该模型,提出了有效的算法。三角形结构聚类不仅能够降低参数对聚类结果的影响程度,还能够发现更加紧密的社区并且能够找到重叠的核心顶点。

当在顶点 4 和顶点 7 之间插入一条新边后,在 C_1 中顶点 4 满足被顶点 3 和 7 三角形结构可达,同时在 C_2 中顶点 4 同样满足被顶点 5 和 8 三角形结构可达,因此 4 是聚类 C_1 和 C_2 的共同核心顶点。在现实的社交网络中,说明 4 是社区 C_1 和社区 C_2 的连接桥梁,是核心中的核心(core in core)。

基于三角形的结构聚类,不仅要满足有边顶点之间的结构相似性要不小于给定的阈值 ε,与此同时,同一聚类中的所有的节点之间必须满足三角形连通的要求,所以本文提出的三角形聚类模型将会产生结构更加紧密的社区。如图 5 所示,当将参数 ε 设置为 0.4, μ 为 2 时基于传统的结构聚类模型会发现聚类 C_1 和 C_2,但是基于本文提出的三角形结构聚类模型,将分别得到更加紧密的社区结构 C_2 和 C_3,节点 4 作为一个核心 Hub 节点连接着这两个聚类,并且这种聚类结果看起来更加合理。

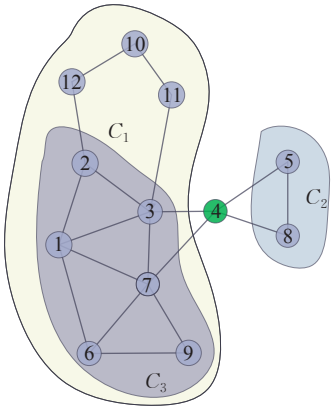


图 5　发现更紧密的社区结构

为了说明三角形结构聚类算法的正确性,给出两个重要的引理,并加以证明。

引理 1　对于给定的图 $G=(V,E)$,如果 $e_m, e_n, e_k \in E$ 顶点间满足顶点间直接结构可达且能够形成一个三角形 $\triangle_1$,那么与 $\triangle_1$ 中任意一条边满足三角形连通,并且满足顶点间直接结构可达的边集将于 e_m, e_n, e_k 形成一个聚类。

证明　(1)非空性:根据假设 e_m, e_n, e_k 可以形成一个聚类,所以 $e_m, e_n, e_k \in C$,因此 C 非空。

(2)最大性:假设 $e_p \in C$ 且 $e_q \in E$,e_p 到 e_q 满足三角形连通且顶点间结构可达:

$\Rightarrow e_m$ 与 e_p 满足三角形且结构可达并且 e_p 与 e_q 满足三角形且顶点间结构可达:

$\Rightarrow e_m$ 与 e_p 满足三角形且顶点间结构可达(根据传递性)

(3)连通性:假设 $e_p, e_q \in C$,有 e_m 与 e_p 三角形连通且结构可达,e_m 与 e_q 三角形连通且顶点间结构可达:

$\Rightarrow e_p$, 与 e_q 通过 e_m 连通

注意:图 G 中一个与参数 ε 和 μ 相关的聚类 C,是由 C 中的任意一条直接可达边决定的,C 中的任意一条边都和其他的边三角形连通。因此,聚类 C 中包含了图 G 中固定的边,且它们之间互相三角形连通且顶点间结构可达。

引理2 对于给定的图 $G=(V,E)$,如果 $C\subseteq E$ 是图 G 的一个聚类,$e_p\in C$ 是 C 的一个直接可达边,那么聚类 C 等价于一个从边 e_p 扩展的一个边的集合。

证明 假设 C' 是通过边 e_p 扩展的一个聚类,只需证明 $C'=C$。

根据 C' 的假设,很容易得到 $C'\subseteq C$;

假设 $\forall e_q\in C$,因为 $p\in C$:

$\Rightarrow\exists e_k\in C$,$e_p$ 与 e_k 三角形连通且 e_k 与 e_q 也三角形连通

$\Rightarrow e_p$ 三角形连通 e_k(根据对称性)

$\Rightarrow e_p$ 三角形连通 e_q

$\Rightarrow e_q\in C$

从而得到 $C'=C$。

通过上述两个引理,本文提出的基于三角形的结构聚类算法的正确性得到验证。接下来,将具体介绍算法的主要过程。

4.2 三角形结构聚类算法

根据上节的基本概述,得到了三角形结构聚类的基本算法过程。

对于给定的图 $G=(V,E)$:

(1)计算得到所以边的结构相似性;

(2)找到满足三角形结构聚类模型的初始边;

(3)从初始边 e 出发,遍历完所以边,输出所有满足条件的聚类。

根据上述主要的过程,得到了三角形结构聚类的主要伪码。在算法3中,首先调用了第二部分计算所有边结构相似性的函数,这里选用基于切片的负载均衡算法(SliceBase函数)并行计算得到所有有边顶点之间的结构相似性,在实验中将简要说明选择基于切片的原因;随后初始化变量 C 存储所有的聚类 C_l,为了得到所有的聚类,需要初始化一个队列 Q 来存储满足三角形连通的边,选中一个满足直接可达的两个顶点,并以这两个顶点之间的边作为出发点进行广度优先遍历不断地扩展聚类,直到队列 Q 为空,则其中一个聚类 C_l 扩展完毕,将其保存在聚类集合 C。反复执行该过程,直到所有顶点划分完成,则算法结束。

算法3 三角形结构聚类

输入:$G=(V,E)$ 图 G,ε 相似性阈值,μ 核心节点判断阈值

输出:图中所有聚类 C

functionTSCAN(ε,μ)

1. Call Function SliceBase to compute σ
2. $C\leftarrow\varnothing$
3. $l\leftarrow 1$
4. $Q\leftarrow\varnothing$
5. for all $v\in V$ do
6. if v is core $\wedge\, v.visited=0$ then
7. $v.visited=1$
8. $R=\{u\in V|DirREACHE(v,u)\}$
9. for all $u\in R$ do
10. $v.visited=1$
11. if $edge_{vu}.visited=0\wedge edge_{vu}.trussness\geqslant 1$ then
12. insert $edge_{vu}$ into Q
13. $edge_{vu}.visited=1$
14. while Q is not empty do
15. $P=\{w\in V|\sigma(v,w)\geqslant\varepsilon\wedge\sigma(u,w)\geqslant\varepsilon\}$
16. for all $w\in P$ do
17. $w.visited=1$
18. if $edge_{vw}.visited=0$ then
19. insert $edge_{vw}$ into Q
20. $edge_{vw}.visited=1$
21. If u is core $\vee\, w$ is core then
22. if $edge_{uw}.visited=0$ then
23. insert $edge_{uw}$ into Q
24. $edge_{uw}.visited=1$
25. insert $edge_{uw},edge_{uw},edge_{uw}$ into C_l
26. $C\leftarrow C_l$
27. $l\leftarrow l+1$
28. return C

5 实验结果及分析

5.1 实验数据

本文实验将使用六个真实的数据集,全部来源于斯坦福大学的官方数据集,并且会选择不同梯度的数据集(http://snap.stanford.edu/data/)。数据集的相关信息如表1所示。

表1 实验数据集

DataSet	Number of nodes	Number of edges
WebGoogle	875 000	4 322 000
Flickr-links	1 715 000	15 551 000
WebBerkStan	685 000	6 649 000
TREC	1 601 000	6 679 000
Skitter	1 696 000	11 095 000
Pokec	1 632 000	22 301 000

5.2 实验环境

本文所有的程序均是用C++语言实现,实验环境为Intel® Xeon CPU E5-2630 v3@2.40 GHz,该服务器具有两个CPU,每个CPU具有八个核心,并且支持超线程技术,内存为32 GB,操作系统为Linux(Ubuntu),支持openMP并行计算框架。

5.3 实验结果与分析

实验将分为两部分:第一部分实验将在不同数目的核心下对比基于切片的负载均衡策略和基于度的负载

均衡策略的运行时间;其次,将在不同参数阈值下对比传统的结构聚类模型(SCAN)与本文提出的基于三角形的结构聚类模型(TSCAN)的聚类效果。

在第一部分实验中,对六个数据集设定统一的相似性阈值(ε)和核心节点判断阈值(μ),并且让CPU的核心数线性增加(2,4,6,8,10)来观察实验结果。

观察图6发现,随着CPU核心数的线性升高,系统的运行时间几乎是线性降低的;另外,根据实验结果可以发现基于切片的负载均衡策略的运行效率普遍高于基于边度($edeg(e_{u,v})$)的运行效率,这也是在算法3中选择调用该函数的原因。这是由于计算有边顶点之间的结构相似性的过程中,当且仅当在完全理想情况下时间复杂度是:$d(u)+d(v)$,但是,由于硬件因素以及算法运行过程中的线程之间的同步问题,从而造成一些误差,使得程序不等达到完全理想的理论运行时间。

然而,基于切片的负载均衡策略是将整个任务均等切分成较小的切片,然后依次将所有的切片分配给各个闲置的CPU核心,只要某个核心完成已有任务,就会立即得到新的任务去执行。并且实验中发现当总的边数超过千万时,切片的大小在1 000到10 000之间较为合适,切片太大会使运行效率降低,更重要的是:通过实验发现,将切片的大小控制在这个范围内,运行时间差距仅仅在1%到2%之间浮动。

接下来,将在具体的实验结果中分析传统的结构聚类模型和本文提出的基于三角形的图结构聚类模型的效果,将取具体数据集中的部分聚类结果进行对比分析,具体如图7、8所示。

以TREC数据集中的某个聚类结构为例,在实验中,首先固定核心节点判断阈值μ,通过改变阈值ε(如图7所示)观察本文提出的三角形结构聚类模型和传统

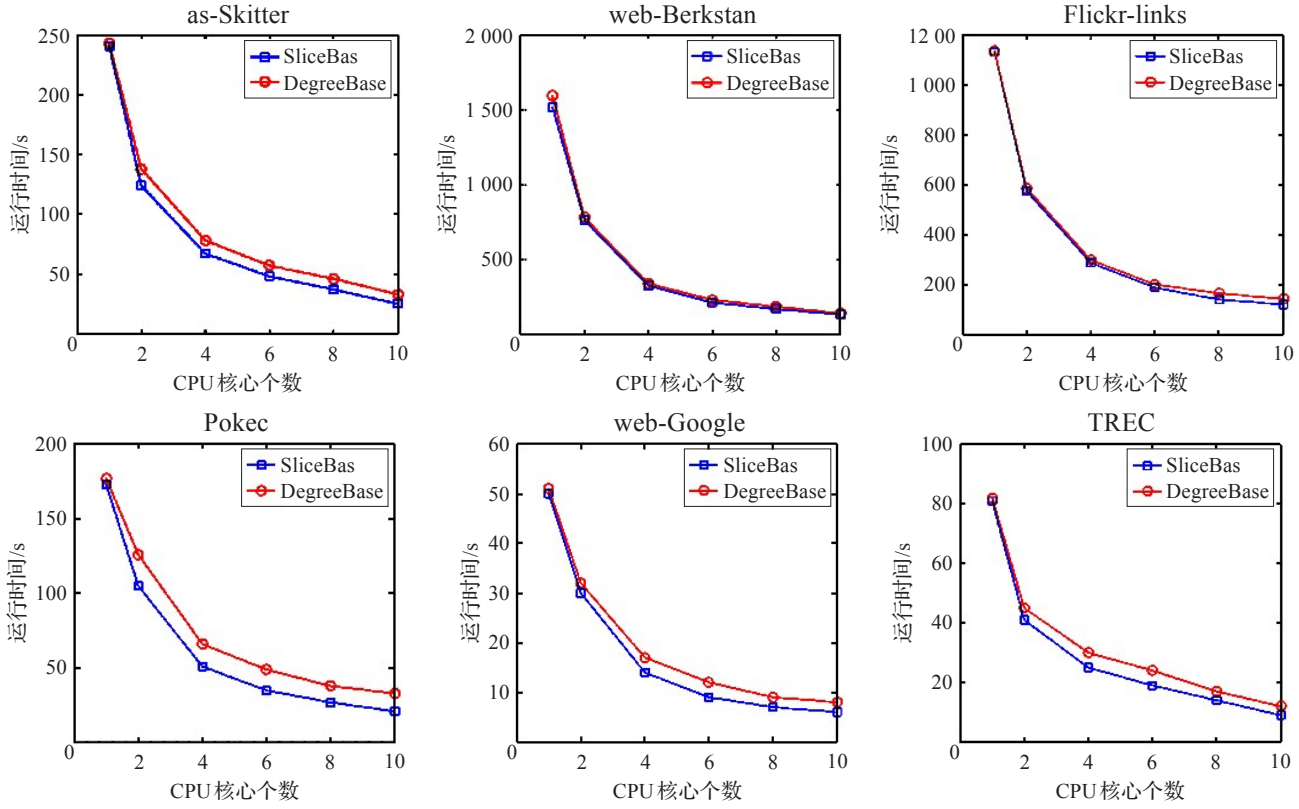
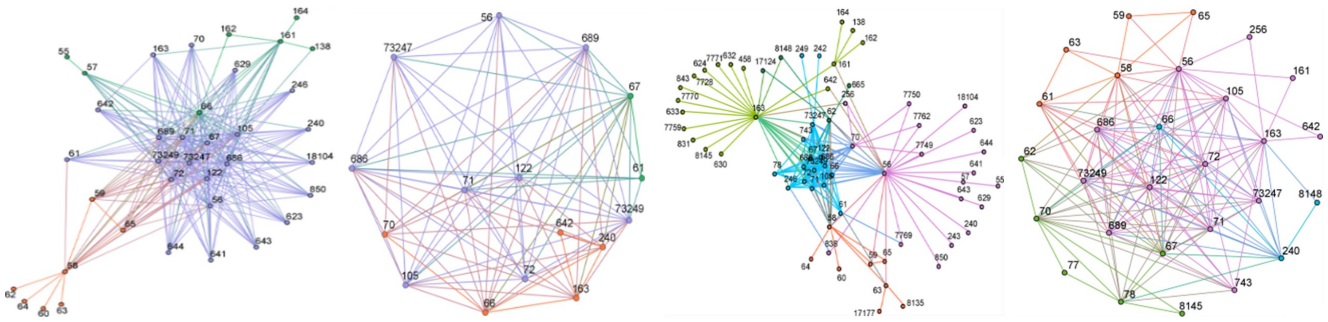
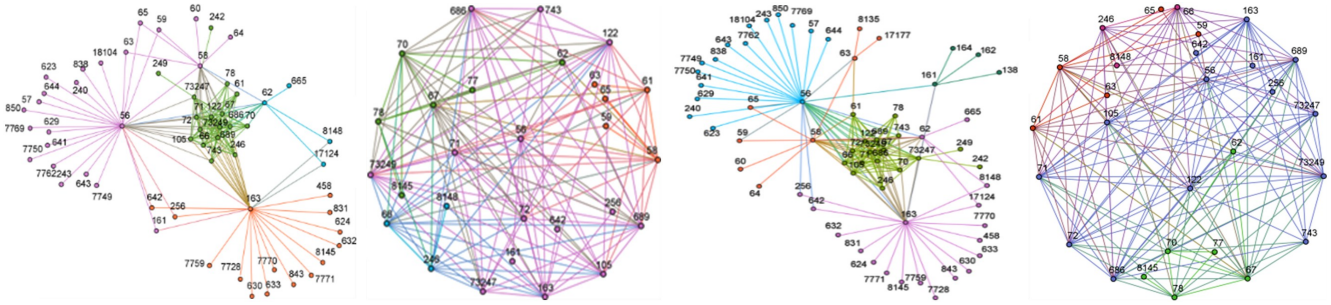


图6　不同负载均衡策略的运行效率对比



(a)SCAN($\mu=2,\varepsilon=0.4$)　(b)TSCAN($\mu=2,\varepsilon=0.4$)　(c)SCAN($\mu=2,\varepsilon=0.35$)　(d)TSCAN($\mu=2,\varepsilon=0.35$)

图7　变化参数ε对比不同模型的聚类结果



(a)SCAN ($\mu=6,\varepsilon=0.35$) (b)TSCAN ($\mu=6,\varepsilon=0.35$) (c)SCAN ($\mu=4,\varepsilon=0.35$) (d)TSCAN ($\mu=4,\varepsilon=0.35$)

图8 变化参数 μ 对比不同模型的聚类结果

的结构聚类模型的聚类效果。在图7中发现阈值相同时,本文提出的基于三角形的结构聚类得到的聚类更加紧密。如图7,基于本文提出的三角形结构(b)共有16个顶点,其所有顶点均在(a)中,并且(a)中有大量的顶点不属于(b),说明本文提出的模型能够得到更加紧密的聚类结构;同时当阈值 ε 降低时,传统的聚类模型对改变比较敏感,改变前后的聚类结果相差较大;而提出的三角形结构聚类模型则相对比较稳定。在图8中,固定参数 ε,改变阈值 μ,发现当阈值 μ 从6降低为4时,基于本文提出的三角形结构聚类模型结构非常牢固,聚类结果并没有发生显著变化。

6 结束语

本文首先在针对传统的结构聚类算法的耗时问题,提出了一种并行化计算边端点之间的结构相似性算法,同时提出了分别基于切片和每条边的度的两种负载均衡策略,并且在实验中都取得了很好的效果。此外,为了解决聚类结果对于相关阈值的过分敏感性,提出了一种基于三角形的结构聚类模型。在实验中,对比了该模型和传统的结构聚类模型的聚类效果,发现本文提出的模型能够得到更加紧密的团体,同时也达到了降低聚类结果对阈值参数的过分敏感的特性,取得了很好的效果。

下一步,将优化基于三角形结构聚类的整个过程,使其能够最大程度并行化,从而提高计算速度;另外,由于参数 ε 以及 μ 需要人工指定,具有很大的随机性和不确定性,不同的参数可能造成聚类结果差别很大,因此将继续深入地进行相关研究,极大地优化这两个参数的选取,比如可以缩小参数的选取范围,使聚类结果更加合理准确;另外,由于现实世界中所形成的图数据总是在不断发生变化不断地进行更新,所以,如何动态的维护已有的聚类显得十分必要。因此,今后也将在这方面展开有关的研究。

参考文献:

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

[2] Jia W,Cheng J.Truss decomposition in massive networks[J].Proceedings of the VLDB Endowment,2012,5(9):812-823.

[3] Huang X,Cheng H,Qin L,et al.Querying *k*-truss community in large and dynamic graphs[C]//Proceeedings of ACM SIGMOD International Conference on Management of Data,2014:1311-1322.

[4] Zhou R,Liu C,Yu J X,et al.Finding maximal k-edge-connected subgraphs from a large graph[C]//Proceeedings of International Conference on Extending Database Technology,2012:480-491.

[5] Ankerst M,Breunig M M,Kriegel H P,et al.OPTICS:Ordering points to identify the clustering structure[J].ACM Sigmod Record,1999,28(2):49-60.

[6] Satuluri V,Parthasarathy S.Scalable graph clustering using stochastic flows:Applications to community discovery[C]//Proceeedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,2009:737-746.

[7] Xu X,Yuruk N,Feng Z,et al.SCAN:A structural clustering algorithm for networks[C]//Proceeedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,2007:824-833.

[8] Shiokawa H,Fujiwara Y,Onizuka M.SCAN++:Efficient algorithm for finding clusters,hubs and outliers on large-scale graphs[J].VLDB Endowment,2015,8(11):1178-1189.

[9] Chang L,Li W,Qin L,et al.pSCAN:Fast and exact structural graph clustering[J].IEEE Transactions on Knowledge & Data Engineering,2017,29(2):387-401.

[10] Cohen J.Graph twiddling in a MapReduce world[J].Computing in Science & Engineering,2009,11(4):29-41.

[11] Park H M,Chung C W.An efficient MapReduce algorithm for counting triangles in a very large graph[C]//Proceeedings of ACM International Conference on Conference on Information & Knowledge Management,2013:539-548.

[12] Suri S,Vassilvitskii S.Counting triangles and the curse of the last reducer[C]//Proceeedings of International Conference on World Wide Web,2011:607-614.

(下转第114页)

的攻击方信息,为网络态势分析与呈现提供依据;另一方面可以根据攻击者信息制定相应防御策略,实现主动防御。

参考文献:

[1] Wang X,Tang H,Paterson A H.Research on the application of firewall in network security[J].Plant Cell,2011,23(1):27-37.

[2] Leu F Y,Tsai K L,Hsiao Y T,et al.An internal intrusion detection and protection system by using data mining and forensic techniques[J].IEEE Systems Journal,2017,11(2):427-438.

[3] Zhang Y,Zhang Y,Zhang Y,et al.Game-theory-based active defense for intrusion detection in cyber-physical embedded systems[J].ACM Transactions on Embedded Computing Systems,2016,16(1):18.

[4] 诸葛建伟,唐勇,韩心慧,等.蜜罐技术研究与应用进展[J].软件学报(自然科学版),2013,24(4):825-842.

[5] Sezer S,Scott-Hayward S,Chouhan P K,et al.Are we ready for SDN? Implementation challenges for software-defined networks[J].IEEE Communications Magazine,2013,51(7):36-43.

[6] Liu X,Hu Z Y.Design and implementation of Web cluster based on Docker container[J].Electronic Design Engineering,2016,24(8):117-119.

[7] Fan W,Fernández D,Du Z.Versatile virtual honeynet management framework[J].IET Information Security,2017,11(1):38-45.

[8] Cohen F.The deception toolkit[EB/OL].(2012)[2017-08-01].http://all.net/dtk/index.html.

[9] Liston L.Welcome to my tarpit:The tactical and strategic use of LaBrea[EB/OL].(2011)[2017-08-01].http://www.hackbusters.net/LaBrea/LaBrea.txt.

[10] 诸葛建伟,韩心慧,周勇林,等.HoneyBow:一个基于高交互式蜜罐技术的恶意代码自动捕获器[J].通信学报,2007,28(12):8-13.

[11] More A,Tapaswi S.A software router based predictive honeypot roaming scheme for network security and attack analysis[C]//Proceeedings of International Conference on Innovations in Information Technology,2013:221-226.

[12] 胡毅勋,郑康锋,武斌,等.Openflow下的动态虚拟蜜网系统[J].北京邮电大学学报,2015(6):104-108.

[13] 邬江兴.网络空间拟态安全防御[J].保密科学技术,2014(10):4-9.

[14] 仝青,张铮,张为华,等.拟态防御Web服务器设计与实现[J].软件学报,2017,28(4):883-897.

[15] 廉哲,殷肖川,谭韧,等.面向网络攻击态势的SDN虚拟蜜网[J].空军工程大学学报(自然科学版),2017(3):79-84.

[16] Subrahmanian V S,Ovelgonne M,Dumitras T,et al.The global cyber-vulnerability report[M].[S.l.]:Springer International Publishing,2015.

[17] 中国国家信息安全漏洞库[EB/OL].[2017-08-01].http://www.cnnvd.org.cn.

[18] Manshaei M H,Zhu Q,Alpcan T,et al.Game theory meets network security and privacy[J].ACM Computing Surveys,2013,45(3):1-39.

[19] Peuster M,Karl H,Rossem S V.MeDICINE:Rapid prototyping of production-ready network services in multi-PoP environments[C]//Proceeedings of Network Function Virtualization and Software Defined Networks,2017.

(上接第83页)

[13] Xiao D,Eltabakh M,Kong X.Bermuda:An efficient mapreduce triangle listing algorithm for web-scale graphs[C]//Proceeedings of International Conference on Scientific and Statistical Database Management,2016:10.

[14] Qin L,Yu J X,Chang L,et al.Scalable big graph processing in MapReduce[C]//Proceeedings of SIGMOD,2014:827-838.

[15] Gabriel E,Fagg G E,Bosilca G,et al.Open MPI:Goals,concept,and design of a next generation MPI implementation[C]//Recent Advances in Parallel Virtual Machine and Message Passing Interface,European Pvm/mpi Users' Group Meeting,Budapest,Hungary,September 19-22,2004:97-104.

[16] Gropp W,Lusk E,Doss N,et al.A high-performance,portable implementation of the MPI message passing interface standard[J].Parallel Computing,1996,22(6):789-828.

[17] Plimpton S J,Devine K D.MapReduce in MPI for large-scale graph algorithms[J].Parallel Computing,2010,37(9):610-632.

[18] Patwary M A.Scalable parallel OPTICS data clustering using graph algorithmic techniques[C]//High Performance Computing,Networking,Storage and Analysis,2013:1-12.

[19] Cazals F,Karande C.A note on the problem of reporting maximal cliques[J].Theoretical Computer Science,2008,407(1):564-568.

[20] Regneri M.Finding all cliques of an undirected graph[R].Seminar—"Current Trends in Ie" Ws,Jun,2007.