基于并行图计算的社区划分方法*

谭敢锋，刘 群†
（重庆邮电大学 计算机科学与技术学院，重庆 400065）

摘 要：进行了以图计算形式划分社交网络社区的调查，在调查中发现如何提升图计算应用于大规模社交网络的计算速度和扩展性，一直是研究的难点。其中针对谱图分割社交网络社区划分进行了研究，为提高谱图划分精确性、解决图计算研究难点，提出了以改进谱图论为基础、并行为实验方式的社区划分方法。实验中利用三角模型，改进了谱图论中谱聚类算法相似权值矩阵构造方法，并结合分布式并行环境 Spark 实现了改进算法并行化。由于三角模型是基于网络结构实际相连关系，反映了网络节点之间的复杂关系，所以通过三角模型计算节点相似性更可靠。通过实际数据集结合并行方式分别与单机分析软件、其他并行算法进行对比实验，提出的并行化图计算方法能有效提升计算速度和扩展性，以及社区划分精确性，支持大规模社交网络的挖掘分析。

关键词：并行图计算；Spark；三角模型；谱聚类
中图分类号：TP301.6 **文献标志码**：A **文章编号**：1001-3695(2018)08-2265-05
doi:10.3969/j.issn.1001-3695.2018.08.006

Community partitioning method based on parallel graph calculation

Tan Ganfeng, Liu Qun†
(*College of Computer Science & Technology*, *Chongqing University of Posts & Telecommunications*, *Chongqing* 400065, *China*)

Abstract: Division of social networking was surveyed in the form of graph calculation, in the survey found that how to improve the calculation and application of graph calculation in large scale social networking, has always been the difficulty. Which in view of the spectrogram segmentation social networking community division was studied, in order to improve the algorithm accuracy, resolve the computing research difficulties, this paper put forward the community partitioning method that based on improve spectrum graph theory, and the experiment based on parallel graph calculation. In the experiments used triangle model, it improved the spectrum graph spectral clustering algorithm in similar weight matrix, moreover combined distributed parallel environment Spark to implement the improved parallel algorithm. Due to the triangle model was based on the actual network structure of linked together, it reflected the complicate relationship between nodes of network, so through the triangular model to calculate the node similarity was more reliable. Through the actual data set, the parallel method was compared with the single analysis software and other parallel algorithm in the parallel condition. The parallelization graph calculation method can effectively improve the calculation speed and expansibility of algorithm, the accuracy of community partition and supports the mining of large-scale social network.

Key words: parallel graph calculation; Spark; triangular model; spectral clustering

0 引言

针对大规模社交网络的划分，聚类方法由来已久，基于谱图理论的谱聚类是目前研究较多、理论深厚且应用较广的聚类算法，该方法利用特征值的谱图结构来降低计算复杂性从而保证聚类质量，相比传统聚类算法更高效。但在数据量较大时，谱聚类计算复杂程度增加，计算量和存储量增大，这些弊端局限了谱聚类在大规模社交网络上的应用。本文提出的并行化方式能更好地解决这类问题，从实验过程和结果可知它带来的不仅是计算速度上的优势，更能体现谱聚类相比于传统聚类方法更具有效性的特点。社交网络结构就是一个巨大网络拓扑结构，根据图论知识可以将网络结构定义为 $G=\langle V, E\rangle$，其中 V 表示各个网络结构顶点，E 表示相连接顶点的边。再由图的数学定义可知，一个图可以用集合来描述，一个点集、一个边集。大规模社交网络拓扑图表示法具有一定局限性，不能合理利用数学知识求得图结构之间的相关特点，获得图结构详细信息。但图的矩阵表示不仅能合理利用数学相关理论求解图特征、图划分等，更能高效地进行图的数据化处理、节约计算资源。目前单机复杂网络分析工具有 NetworkX、igraph 等，NetworkX 在效率上没有 igraph 高效，NetworkX 程序结构比 igraph 更具优势，两者包含了图论的所有方法，具有适合分析人员自建模型的功能，但在大规模复杂网络下，这些分析工具显得独木难支，对单机 PC 性能要求更是苛刻。并行环境下处理分析大规模复杂网络的工具有 GraphLab、GraphX 等，GraphLab 基于 MapReduce 进行计算，GraphX[1] 基于 Spark 进行内存计算，GraphLab 底层使用 C ++，而 Spark 底层使用 Scala 编写需要基于 Java 虚拟机环境，所以在并行化分析大规模复杂网络时 GraphLab 效率明显高于 GraphX。而图结构可用矩阵表示，这样 Spark 内存计算比 Hadoop 的 MapReduce 更具优势，因此比单一的 GraphLab 图计算性能更佳。

本文提出基于 Spark[2] 内存并行分布式矩阵计算的方式，对改进的谱聚类算法实现并行，然后对大规模复杂社交网络进行分析，实现了并行化图计算[3]。通过对谱聚类算法进行改进对图结构进行划分，可以得到本文提出的方法在扩展性、高效性、正确性上更具优势。利用真实的大规模复杂网络数据与其他模型作对比实验，然后进行社交网络模块度检测评价[4]，综合表明本文提出的方法在社区划分应用中有效性较高，扩展性较强。实验使用数据来源于 Stanford large network dataset

收稿日期：2017-04-03；**修回日期**：2017-05-26 **基金项目**：中国重庆研究生科研创新项目(CYS16161)；中国国家自然科学基金资助项目(61572091)；重庆自然科学基金资助项目(CSTC2014jcyjA40047)；CQUPT 博士研究生创业项目((A2014)-20)

作者简介：谭敢锋(1991-)，男，重庆人，硕士，主要研究方向为数据挖掘、大数据；刘群(1969-)，女(通信作者)，重庆人，教授，主要研究方向为数据挖掘、智能信息处理、复杂网络(liuqun@cqupt.edu.cn).

collection,各实验包含大规模级顶点和边。

1 谱聚类基本理论和算法描述

1.1 图划分原理

图划分基本要求为,图之间相似度低,子图内相似度高[5]。寻找图最优划分,需要建立划分准则,在图论中常见的划分准则有 N-cut、Ratio-cut、M-cut 等,划分准则的好坏直接影响到最后聚类的优劣。鉴于文章篇幅,在这里主要介绍 N-cut 和 Ration-cut 划分算法。

a)规范化割集准则(normalized cut),由 Shi 等人提出新的目标代价函数为

$$\mathrm{Ncut}(A,B)=\frac{\mathrm{Cut}(A,B)}{\mathrm{Vol}(A,V)}+\frac{\mathrm{Cut}(A,B)}{\mathrm{Vol}(B,V)} \tag{1}$$

其中:$\mathrm{Cut}(A,B)=\sum_{i\in A,j\in B}\omega_{ij}$意为 A、B 两个子图,$v_i\in A$,$v_j\in B$ 两个顶点之间边相似权值之和最小;$\mathrm{Vol}(A,V)=\sum_{i\in A,t\in V}\omega_{it}$ 意为 $A\cup B=V$,$\mathrm{Vol}(A,V)$属于子图节点的相似度权值之和。这样就平衡了样本间和样本内的相似度,避免了小区域分割。

b)比例割集准则(ratio cut),由 Kahng 等人提出的 Ratio-Cut 函数为

$$\mathrm{RCut}(A,B)=\frac{\mathrm{Cut}(A,B)}{\min(|A|,|B|)} \tag{2}$$

在该函数中,RCut 引入了一个规模参数作为分母,$\min(|A|,|B|)$表示 A、B 中顶点的个数,这样就加大了子图的差异性,避免了过分分割,但划分效率较低。

1.2 谱聚类基本原理及算法

由图论思想衍生的谱聚类是一种聚类方法[6],谱聚类算法将聚类问题用图划分的思想表示,通过子图的划分来确定聚类群体。谱聚类适合任意形状数据集,可以获得全局最优解,算法中心思想在于对数据构造拉普拉斯矩阵(Laplacian matrix),然后通过对拉普拉斯矩阵的特征分解获得特征向量进行聚类。

在谱图理论使用中一般基于 NCut 的谱聚类算法使用较为广泛,以下将基于 NCut 分割进行算法的描述。

算法 1 基于 NCut 的谱聚类算法

输入:数据集 S,输入需要划分的 k。

输出:数据集 S 的 k 个子图划分。

根据社交网络结构数据,构造图结构下的邻接矩阵 W;

计算度矩阵 D,构造 Laplace 矩阵,$L=D-W$,其中:

$D=\mathrm{diag}(d_1,d_2,d_3,\cdots,d_n)$,$d_i=\sum_j^n W_{ij}$;

将非规范 Laplacian 矩阵转换为规范化 Laplacian 矩阵,

$L=I-D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$;

计算 $Lv=\lambda DW$ 前 k 个最大特征值对应的特征向量

$v_1,v_2,v_3,v_4,\cdots,v_k$;

以 $v_1,v_2,v_3,v_4,\cdots,v_k$ 为列向量构成矩阵 $V\in\mathrm{R}^{n\times k}$;

令 $y_i\in\mathrm{R}^k$,$i=1,2,3,\cdots,n$ 是 V 的第 i 个行向量;

将原来是 $n\times n$ 的矩阵,用这 k 个列向量降维为 $n\times k$ 矩阵;

用 K-means 算法对上式进行聚类,得到 $c_1,c_2,c_3,\cdots,c_k$;

得到数据集 k 个聚类划分结果。

2 基于三角理论构造近邻图

在社交网络群体中,三角关系是最基本的结构,依据社会平衡理论“朋友的朋友是朋友”[7],对于无向网络,这是最简单的网络结构基础。利用节点到相邻节点相似性来反映节点之间的相似关系,根据相似权值来确认节点之间的紧密程度,更能反映数据节点之间的近邻可靠性,这样构造的近邻图更能反映网络结构。基于以上条件,提出了基于三角理论构造近邻图的方法,首先基于原始的无向稀疏网络结构,遍历出每个节点的相邻集合点,然后根据交集方式,确定共有集,进而通过概率计算获得相似权值,构造节点间稀疏权值矩阵。具体步骤如下:根据相似度度量函数,计算各节点间相似度,构造相似图,构造权值邻接矩阵 ω。

2.1 三角模型建立

给定网络结构数据集,$S=\{x_1,x_2,x_3,\cdots,x_n\}$,$x_i\in V$。首先计算相邻节点之间的相似性:

$$S_{ij}=\begin{cases}\dfrac{V_i\cap V_j}{S} & i\neq j\\ 0 & i=j\end{cases} \tag{3}$$

其中:V_i、V_j 表示各个节点相邻的节点集,节点本身定义为零。

利用网络数据结构集 S,根据节点相似性三角模型,可以获得一个带权值相似图 $G=(V,E,W)$,V 代表网络数据集中有效节点集,E 代表相似节点之间的边,W 代表相似节点间的权值。以此可过滤孤立点和对社区贡献率较少的点,截断相似性较低的边,获得新网络采样图如图 1 所示。

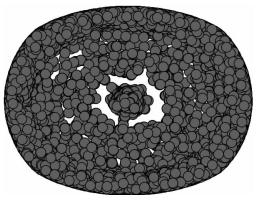

图1 采样图

2.2 近邻点选择

建立三角模型后,使用中位数计算法确定有效的节点度,同样使用中位数计算法确定有效权值。由于度符合长尾理论,这样某些点对应权值和度取值较大,为了计算的平衡性使用以上方式确定合理取值点,度长尾图如图 2 所示、权值分布图如图 3 所示。

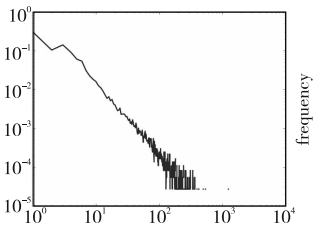

图2 度分布图

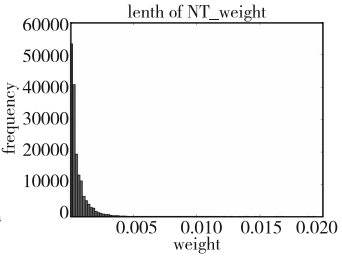

图3 权值分布图

在图 2 中为了详细表示长尾理论,横坐标指数位减少到原数据的一半;图 3 中不仅表示了权值分布,另一方面也表示了社交网络邻接图的稀疏性。将基于三角理论的近邻图应用于谱聚类,只需要将上面算法描述中图的邻接矩阵 W 转换为三角模型相似权值矩阵。三角模型求解节点间相似性,一方面避免了距离求解相似性的不确定性,另一方面避免了基于随机游走求解相似性的随机性。通过实验表明提出的三角模型,避免了像其他方法在求解相似性时需要人工参与调节参数的情况,该方法完全基于网络结构特征,避免人为干扰实验精度等问题。虽然是简单改变了节点相似性权值的求法,但却提出了完全基于网络结构特征的求解方式,更符合图划分的特点,即图之间具有较弱相似性,图内具有较强相似性的特点。改进的谱聚类并行化社区划分算法描述如下。

算法 2 改进的谱聚类并行化算法

输入:数据集 S,输入需要划分的 k。

输出:数据集 S 的 k 个子图划分。

根据社交网络结构数据,构造图结构下的邻接矩阵 W;

计算度矩阵 D,其中

$D=\mathrm{diag}(d_1,d_2,d_3,\cdots,d_n)$,$d_i=\sum_j^n W_{ij}$;

根据并行化三角模型,获得节点间相似权值;

规范化 Laplacian 矩阵,$L=D^{-\frac{1}{2}}\omega D^{-\frac{1}{2}}$;

并行化计算 $Lv=\lambda D\omega$ 前 k 个最大特征值对应的特征向量 $v_1,v_2,v_3,v_4,\cdots,v_k$;

以 $v_1,v_2,v_3,v_4,\cdots,v_k$ 为列向量构成矩阵 $V\in\mathrm{R}^{n\times k}$;

令 $y_i \in R^k, i = 1,2,3,\cdots,n$ 是 V 的第 i 个行向量;
将原来是 $n \times n$ 的矩阵,用这 k 个列向量降维为 $n \times k$ 矩阵;
用并行化 K-means 算法对上式进行聚类,得到 $c_1, c_2, c_3, \cdots, c_k$;
得到数据集 k 个聚类划分结果。

2.3　基于 Spark 的三角模型并行化

由于基于三角模型求解节点相似性权值,最坏情况下时间复杂度为 $O(n^{n-1})$,为了解决时间复杂度较高问题,本文提出了并行化方式。既利用 Spark 平台并行化及基于内存计算优势[8],又利用分而治之的思想,解决此问题。处理模型如图 4 所示。

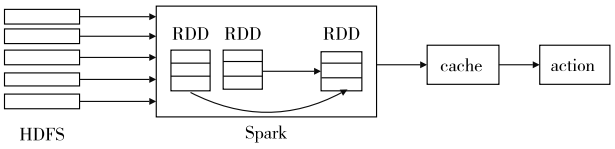

图4　并行三角模型实现过程

在构造基于 Spark 三角模型时,将原始数据上传于 HDFS (Hadoop distributed file system)[9],该文件系统具备高容错、高吞吐特征,不仅保证数据安全性问题,而且更能保证数据完整性,作为 Spark 计算可靠数据来源之一。当 Spark 获得请求任务处理时,Spark 会在心跳机制下获得当前集群中 alive work, Spark 同时会根据 work 给 master 汇报的资源信息,结合 Application 提出的应用请求,合理分配当前资源,Spark 遵从于粗粒度分配,使系统能够最大化服务于请求。在 application 启动时会运行 runJob( )方法,启动 DAGScheduler 来划分 job 的 stage,同时启动 taskSchedulerImple 负责 stage 内部底层调度;然后接收到 launchTask 消息启动 coarseGrainedSchedulerBackend 来调用 coarseGrainedExecuterBackend 实例化其中的 executor;最后由 executor 下唯一对应的 task 执行任务。为了提高运算速度,在最终执行 action 触发 job 前将之前结果缓存于内存之中,获得计算结果为节点之间的相似权值。

Spark 计算中遵守“数据不动代码动”原则,远程端代码提交对 HDFS 上网络节点进行 map 操作,确定相同源点的终点 map 集合,转换为 kv 形式;然后通过 filter 对 map 集下转换而来的 RDD 进行交集过滤,转换为下一步需要计算的 RDD;最后,执行 action 操作,产生 shuffle 过程将各个分布式节点汇聚到 master 节点中,将结果数据集持久化于 HDFS 中,同时缓存作为谱聚类权值计算使用。

以上过程中,在 action 之前主要的工作在于标记算子和确认数据,没有进行真正的计算,这样也为其他任务的提交留足了资源。当在执行 action 操作时,所有的算子进行管道式计算,这时需要注意的是系统性能问题,可能发生数据倾斜、内存溢出、计算时间过长等问题。为了解决这类问题,应尽量将内存增大的同时,考虑在发生数据倾斜时是否是计算过程中某个 map$\langle k,v\rangle$数据过于庞大的情况,针对该类问题一般添加随机数的方式将发生倾斜的数据分散处理。Spark 计算时间的长短在于最后计算完的那个 task,如果计算过程中时间过于长,除了考虑数据倾斜,同时需要考虑数据本地性是否过大等原因。由于 Spark 基于 lineage 的高效容错性,不需要担心计算过程中发生错误导致整个计算崩溃,在血统机制下会自动将发生错误的算子进行重新计算,所以提高了本文提出的三角模型的稳定性。

3　基于 Spark 的改进谱聚类模型并行实现

对于改进的谱聚类主要有三个步骤:由并行三角模型构造 Laplacian 矩阵,计算 k 个特征值及特征向量,利用并行 K-means 进行聚类。在并行三角模型计算结果缓存数据的基础上,构造 Laplacian 稀疏矩阵,根据数据本地性,将每个分布式并行节点计算中社交网络节点相似性权值构造的 map$\langle k,v\rangle$键值对转换为 RDD,为后面算子构造矩阵提供数值,构造过程如图 5 所示。同时对每个 RDD 中数值长度进行计数,作为度矩阵 D 中的数值。谱聚类并行算法的实现过程如图 6 所示。

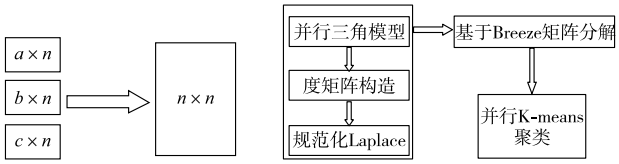

图5　并行分布式矩阵构造过程　图6　改进谱聚类算法并行实现过程

3.1　并行三角模型构造 Laplacian 矩阵

并行三角模型构造 Laplacian 矩阵,主要有以下步骤:首先根据三角模型,计算有效度节点相邻节点相似度权值,构造权值矩阵;然后,根据有效度节点,构造度矩阵;最后,由度矩阵和相似度矩阵,构造规范化的 Laplacian 分布式矩阵。在这几步中,需要确定有效度,由于社交网络节点符合长尾理论,在这里求解了度集合的中位数,以此确定大于或者等于中位数度节点为社交网络结构中的有效节点。在构造度矩阵和相似度矩阵时,在集群中每个计算节点中的分布式矩阵行数不相等,如图 5 所示,a、b、c 大小不一定相等。在整个过程中涉及集群多机协同工作,整个计算的快慢取决于集群中完成计算任务的最后一个计算节点。为了提高计算速度,避免产生内存泄露,在这里借助了第三方集群监控软件进行监控,对并行化过程进行实时监控。

构建过程为,在三角模型计算权值 pipeline 的基础下将源节点和目标节点转换为新 map 集 RDD,形如获得三元组形式也就是 Spark 下的坐标矩阵形式 coordinateMatrix,将坐标形似的每个元素转换到 Spark 分布式矩阵支持的数据类型 long;然后调用 matrixEntry 方法将坐标矩阵构造成带权值参数,同时生成新的 mapPartitionsRDD,实例化 coordinateMatrix 构造真正的带权值分布式矩阵的 RDD,为后面计算特征值作准备。

3.2　基于 Breeze 的矩阵分解

改进的谱聚类算法构建邻接矩阵,用于计算谱聚类算法下的特征值和特征向量。Spark 非常擅长内存迭代计算,一方面避免了单机计算物理设备限制,另一方面提高了运算速度。由于是基于 Spark 的矩阵运算,它唯一支持的线性运算库为 Breeze,它提供了矩阵运算的各类方法函数,本实验对其进行了相应调用,快速获得了结果。在这部分计算中需注意的主要有两步:

a)在规范化 Laplacian 时,存在某些节点非有效节点,这样使权值相似矩阵的行存在全零值的情况,为了不改变谱聚类的算法,利用行对应度矩阵中度为零,规范化乘积结果不发生变化的特点,为该行添加一个有效相似权值。通过分析对比中位数法和众数法的计算结果发现中位数具有有效性,众数容易受到局部大数的影响,数值不具有效性。

b)改进的谱聚类算法,主要求解较大特征值和对应的特征向量,为了尽量减少人工参与,本实验选择特征值时使用启发式的方式获得 k 个最大值。

计算分布式矩阵特征值,基于三角模型 pipeline 下分布矩阵 RDD,调用 Spark 线性运算库 Breeze 下 eig 方法,执行 Spark 的 action 操作 shuffle 各个分布式矩阵计算结果,得到分布式矩阵特征值计算结果。将特征值计算结果持久化 HDFS,利用可视化工具 Pyplot 得到特征值计算结果趋势图。

3.3　并行 K-means 聚类

基于 Spark 的并行 K-means 算法,该算法封装于 SparkMLlib 中,为 K-means 并行化带来了极大的便利,当 Breeze 计算出 k 个特征值和对应特征向量后就将 $n \times n$ 的矩阵降维到 $k \times n$ 的矩阵。再选取 k 个对象作为初始聚类的中心,把初始中心进行迭代,在每次迭代中对数据集中剩余的每个对象与各个簇中心的距离进行计算,将每个对象重新赋给最近的簇。当考察完所有数据对象后,一次迭代运算完成,新的聚类中心被计算出来,以此往复。如果在某一次迭代前后,聚类中心的值没有发生变化,说明算法已经收敛。对于 K-means 算法,其本身的时

间复杂度为 $O(NKt)$，其中 N 是数据对象的数目，t 是迭代的次数，k 为聚类数。在基于 Spark 并行化下时间复杂度为 $O(NKt/$集群节点数$)$，这样计算时间得到了相应的减少，同时 Spark 基于内存计算提高了计算速度。

4 实验分析

本实验主要验证改进谱聚类算法有效性、并行谱聚类算法的高效性和并行谱聚类算法的可扩展性。本实验的数据来源于 Stanford large network dataset collection，其中实验设备为三台 HP 服务器，2 GHz 的 CPU、16 核、16 GB 内存。本实验针对算法有效性使用了基于模块度的测量标准。对于并行谱聚类算法的高效性和可扩展性以实验对比的方式进行说明。

4.1 改进的谱聚类算法有效性

针对算法有效性，在本实验阶段通过与其他算法作对比实验和模块度(modularity)检测两种方式进行综合评价。算法上主要与随机游走(walktrap)社区划分算法[10]、基于模块度(modularity)[11]社区发现算法、infomap 算法[12]作对比。模块度检测算法是社交网络中检测社区划分后社区聚合稳定性的重要指标。

模块度检测算法公式如下：

$$Q = \sum_c e_c - e_c^2 \tag{4}$$

其中：Q 值为所有划分社区模块度之和，在计算结果中 Q 值越接近 1 代表社交网络社区划分稳定性越好，划分效果越突出。e_c 代表划分社区内所有边权值之和，代表社区之间边权值平方。改进的并行谱聚类算法有效性与上述算法对比如表 1 所示。

表 1 改进谱聚类算法有效性对比

算法	模块度
walktrap	0.369 365
基于模块度社区发现	0.370 269
infomap	0.398 299
改进的并行谱聚类	0.398 753

前三个算法基于目前运行效率最高的 igraph 图计算软件，对于改进的并行谱聚类算法则是基于 Spark 平台。从表 1 中可以知道在 Stanford large network dataset collection 测试数据集 email-Enron 中，各个算法对应的模块度大小。改进的并行谱聚类算法，得到的结果明显高于随机游走和基于模块度社区发现。相比较于 infomap 的算法，改进的并行谱聚类几乎两者模块度度量相等，但相较于前两个算法有效性更好，改进后的谱聚类有效性得以很好的体现。

4.2 并行谱聚类算法的高效性

为了不在设备上产生差异性，选择相同服务器作为实验环境。同样使用 Stanford large network dataset collection 测试数据集 email-Enron，不同算法运行时间如图 7 所示。

从图 7 可以知道，当节点数较多，各个算法在不受设备环境的影响下，运算时间相差较大。在 igraph 中运行 walktrap 算法时，整个计算的过程大约 12 h；在执行基于模块度社区发现算法和 infomap 算法时，计算时间分别是 9 h 和 7 h 左右；但在改进的并行谱聚类方法下计算时间约 0.2 h，改进的谱聚类算法在并行下的高效性获得了很突出的表现。

为了尽量减少人为参与选择改进谱聚类算法求解最大特征值个数，使用启发式选择特征值，特征值变化如图 8 所示。

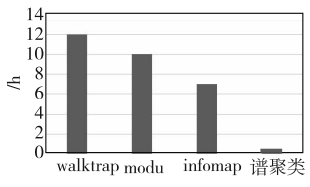

图7 不同算法高效性对比

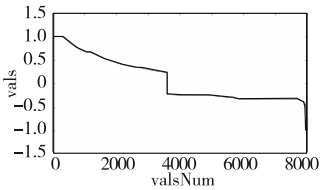

图8 特征值分布图

使用 Breeze 计算矩阵特征值，特征值如图 8 所示，当计算到约 3 000 个特征值时，特征值呈现断崖式下降，当计算 8 000 个特征值时又呈现了极速下降趋势。在谱图理论下，求解规范化 Laplacian 特征值最大 k 个特征值，求解个数依据最大特征值趋势图，特征区间在 0.3 ~ 1，该区间下特征向量有 3 000 个左右，如果在这里直接进行 K-means 聚类会产生最小团问题，导致图过分划分。为了解决过分划分，确定迭代聚类中心，选择聚类中心趋于稳定的 k 个值，使能够合理选择最大 k 值。将 0.3 ~ 1 以 0.05 的递增顺序增加，得到 15 个这样的特征值，谱聚类下依据最大特征值个数确定划分个数[13]，对选择 k 进行迭代获得肘变程度图，如图 9 所示。肘部法则是判断聚类个数的常用方法，其成本函数表示为

$$J = \sum_{k=1}^{k} \sum_{i \in c_k} |x_i - u_k|^2 \tag{5}$$

肘变程度等于聚类中心与其内部各个成员距离差值的平方和，u_i 为聚类中心的位置。由于本文使用相似性判断节点间距离，在这里将距离改变成聚类点到每个节点之间的相似度大小，u_k 为聚类内部相似权值最大值，同时将计算结果的数值扩大 10 倍，便于对实验结果的肘变程度绘图。

通过图 9 可以得到，当 k 在 2 左右时产生肘变图，这时的划分数对整个谱图聚类影响较大，在聚类个数上聚类中心迭代较为稳定。用并行的 K-means 聚类方法，对降维后由 k 个特征值组成的特征向量矩阵数据进行聚类，得到的社交网络划分结果如图 10 所示。在 4.1 节中使用了其他三个参考模型作对比，评价聚类性能，改进后的谱聚类总体上聚类效果趋于上游，整体上满足聚类内部相似较高，聚类之间相似较低原则，但是并行的加入大大提高了算法运算速度，得到了改进的谱聚类在整体上优于其他参考模型。从聚类结果图 10 中分析可得，划分结果符合实际情况，源数据信息表示为公司内部的邮件来往，整个公司内部邮件往来是基层和上层两个大类别，同级之间上层聚类类间数相较于基层聚类类间数目要低得多，根据可视化聚类结果，改进的谱聚类算法聚类结果符合实际情形。

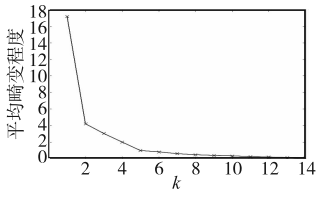

图9 肘部法确定k值

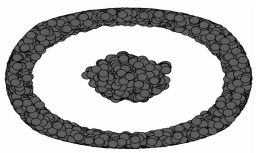

图10 社交网络划分结果

4.3 改进的谱聚类算法并行下的可扩展性

在对改进的谱聚类算法并行化下的可扩展性进行测试时，分别与不同算法在不同规模网络下进行对比实验[14,15]。数据来源于 Stanford large network dataset collection 测试数据集和 Twitter 数据集，整个过程选取三个不同规模进行对比实验，无向图数据集 1 为 36 692 个点，183 831 条边；无向图数据集 2 为 334 863 个点，925 872 条边；无向图数据集 3 为 65 608 366 个点，1 806 067 135 条边。实验结果如图 11 所示。

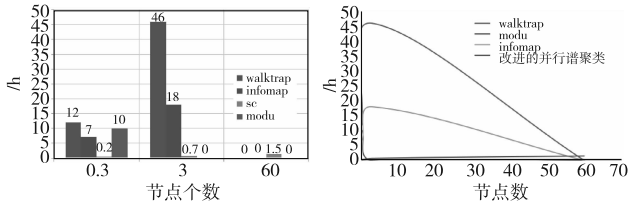

图11 改进谱聚类并行化可扩展性对比

图中横坐标为节点个数，为了将差异表示清晰，这里将节点用两位数值表示。从图可知当节点在 3 万左右时，walktrap 算法计算时间为 12 h 左右，infomap 算法计算时间为 7 h，模块度算法计算时间为 0.2 h；当节点在 30 万左右时，各个算法计算时间相应增加，而模块度算法却突然转变为 0，这是由该算法复杂度导致计算无法进行计算，程序无法运行；当节点在 6 千万左右时，由单机扩展性限制和各算法复杂性导致计算无法运行，单机直接报内存错误，为了判断是否是单机设备导致

算法无法运行,将设备内存扩大到 32 GB,选择单机执行效率最高的 infomap 算法进行测试,算法无法运行,程序自动终止,相反,改进的谱聚类算法并行化下虽然处理时间增加,但稳定性上高于单机处理,同时处理大规模数据范围较大,可扩展性上强于对比算法。在大数据集下,为了排除只是处理规模优势,在这里对单机下可处理的数据作模块度比较,确定改进的谱聚类算法并行化在大数据集同样具有有效性,当数据集是 30 万左右的节点集时,模块度检测如表 2 所示。

表 2　改进谱聚类算法有效性对比

算法	模块度
walktrap	0.239 897
infomap	0.304 325
改进的并行谱聚类	0.295 341

在大规模社交网络实验对比下,改进的谱聚类算法并行化有效性较高,相比较于经典的 infomap 算法,约低于 0.008 左右模块度,但结合处理速度和可扩展数据集大小,改进后的谱聚类算法并行化下优势明显。

4.4　并行环境下改进的谱聚类算法与并行化 Louvain 对比

这里利用基于 Louvain 社区划分并行算法与本文提出的算法进行对比,主要对比为算法效率、算法划分准确性,Louvain 算法是基于模块度的社区发现算法,其优化目标是最大化整个社区网络模块度。

Louvain 算法:

a)将社交网络每个节点看做一个单独的社区,社区数目与节点个数相同;

b)每个节点 node 与其相连节点社区进行融合,并计算融合前与融合后模块度变化,记录模块度变化最大的那个 node,如果该模块度大于零则将此节点与模块度变化最大节点划分为同一个社区;

c)重复步骤 b),直到所有节点所属社区不再改变;

d)将同在一个社区的节点压缩成一个新的节点,新节点之间的边权重为社区之间权重值;

e)重复步骤 a)直到整个图模块度不再改变;

通过并行环境下分别实现本文提出的改进谱聚类和 Louvain 算法,实验结果如表 3 所示。

表 3　改进的谱聚类与 Louvain 对比

算法	算法效率	划分准确性	最优性
改进的谱聚类	高效	较高	全局最优
Louvain	高效	人工参与下较高	局部最优

由于原始的谱聚类无法进行大数据下的运算,通过改进和实现并行化解决了这一问题,在并行化环境进行比较时,谱聚类可以得到全局最优,无须进行多次重复计算,但是 Louvain 算法容易得到局部最优,需要进行多次重复计算,取结果基本平衡时为最终划分结果。改进的谱聚类划分结果通过模块度检测,但 Louvain 算法很多时候是通过人工干预获得计算准确性;在最优性上改进的谱聚类强于 Louvain 算法,实现全局最优。在分布式 Louvain 实验中,在解决消息延迟和节点重复划分问题方面,参考了某购物网站技术部解决方案,同时对其进行了改进,解决问题主要在于,划分分片时存在一个节点被划分为两个分片中,在这个阶段,在实验中记录了这些公用节点,并广播这些节点信息,当划分社区包含该节点,社区压缩为同一个节点时以公用节点标志,在实验中每个分片获得广播信息后会对分片无用信息进行删除以降低内存开销。

5　结束语

以图计算方式处理大规模社交网络数据,存在效率较低、受单机设备限制的问题。结合并行及改进的谱聚类算法,能有效改善此类问题。通过实验结果可以得到,该方式下具有较高的准确度,并行化下高效性得以很好的体现。Spark 并行下既具有高速的计算能力,又能将中间结果缓存于内存中从而减少大量的 I/O 操作;同时基于血统机制,对计算失败等情况不需要重新计算,大大改善了运算过程。当结果需要以稳定的持久化方式进行存储时,依靠 HDFS 的高吞吐和高容错能力,能很好地保证数据的完整性。改进后的谱聚类算法,直接依靠网络结构来计算节点间的相似性,减少人为判断参数,提高相似节点的可信性。同时,结合 Spark 分布式并行处理,在克服算法复杂性的基础上,运算速度和准确性得以提升。以图思想解决大规模复杂网络划分问题,有效地用矩阵的形式表示图结构,再转换为数值计算。实验过程的设计,一方面考虑并行方式的运用,另一方面还注重算法有效性、高效性、并行下可扩展性的实验对比和分析。在有效性的分析中,通过与常规方法进行对比,数据表明该算法更具有效性。在高效性分析中,通过与其他算法进行处理时间对比,得出改进后的算法更具高效性。在扩展性分析中,不仅与其他算法进行对比,同时使用不同规模的数据进行验证,得出该并行方式下可扩展性强的结论,适合较大数据规模。

通过对比实验,本文提出的算法具有明显高效的优势。在大数据环境下,本文提出的基于并行图计算的社区划分方法,既能对大规模社交网络进行分析,又对高维数据并行化处理有一定的借鉴作用。

参考文献:

[1] Xin R S, Gonzalez J E, Franklin M J, *et al*. GraphX: a resilient distributed graph system on Spark[C]//Proc of the 1st International Workshop on Graph Data Management Experiences and Systems. New York: ACM Press, 2013: Article No. 2.

[2] Han Zhijie, Zhang Yujie. Spark: a big data processing platform based on memory computing[C]//Proc of the 7th International Symposium on Parallel Architectures, Algorithms and Programming. Piscataway, NJ: IEEE Press, 2016: 172-176.

[3] Bourdonov I B, Kossatchev A S. Parallel computations on a graph [J]. Programming & Computer Software, 2015, 41(1): 1-13.

[4] Newman M E J, Girvan M. Finding and evaluating community structure in networks[J]. Physical Review E: Statistical Nonlinear & Soft Matter Physics, 2004, 69(2): 026113.

[5] Ghoshal G, Zlatić V, Caldarelli G, *et al*. Random hypergraphs and their applications[J]. Physical Review E: Statistical Nonlinear & Soft Matter Physics, 2009, 79(6): 853-857.

[6] Alzate C, Suykens J A K. Sparse kernel spectral clustering models for large-scale data analysis[J]. Neurocomputing, 2011, 74(9): 1382-1390.

[7] Heider F. The psychology of interpersonal relations[M]. [S. l.]: Wiley, 1958.

[8] Antoine H, David A. Distributed graph layout with Spark[C]//Proc of International Conference on Information Visualisation. Washington DC: IEEE Computer Society, 2015: 271-276.

[9] Ghazi M R, Gangodkar D. Hadoop, MapReduce and HDFS: a developers perspective[J]. Procedia Computer Science, 2015, 48: 45-50.

[10] Meila M, Shi Jianbo. A random walks view of spectral segmentation [C]// Proc of the 8th International Workshop on Artificial Intelligence and Statistics. 2001.

[11] Newman M E. Modularity and community structure in networks[J]. Proceedings of the National Academy of Sciences, 2006, 103(23): 8577-8582.

[12] Lancichinetti A, Fortunato S. Community detection algorithms: a comparative analysis[J]. Physical Review E: Statistical Nonlinear & Soft Matter Physics, 2009, 80(2): 056117.

[13] Cao Jiangzhong, Chen Pei, Dai Qingyun, *et al*. Local information-based fast approximate spectral clustering[J]. Pattern Recognition Letters, 2014, 38(1): 63-69.

[14] Sakr S. Large-scale graph processing systems[M]//Big Data 2.0 Processing Systems. [S. l.]: Springer International Publishing, 2016: 53-73.

[15] Benson A R, Gleich D F. Higher-order organization of complex networks[J]. Science, 2016, 353(6295): 163-166.