# 一种基于局部拓展的并行重叠社区发现算法

张忠正 李建武
(北京理工大学计算机学院 北京 100081)

**摘 要** 处理海量级数据的有效途径之一是将算法分解为一系列互不依赖的任务,然后利用开源工具并行地执行算法。而在重叠社区发现算法中,基于局部拓展的方法在拓展阶段往往仅需要局部社区及其相应的邻居结点的信息,因而具备可并行执行的可能性。提出了一种可并行化执行的局部拓展算法,并借助开源工具 Spark 将其实现。算法分为4个阶段。首先,挑选出一组不相关的中心结点并使用其对应的局部网络作为种子;其次,通过删除本身连接不是很紧密的局部网络来过滤选出的种子;然后,采用一种批量式的拓展策略来拓展种子,即一次向局部社区中添加一批邻居结点或从社区中删除一批结点;最后,融合相似的社区。在人工生成的网络以及真实世界中的网络上的实验结果显示,所提算法既准确又高效。

**关键词** 复杂网络,重叠社区发现,局部拓展,并行化算法,Spark
**中图法分类号** TP181 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.9.011

## Parallelizable Overlapping Community Detection Algorithm Based on Local Expansion

ZHANG Zhong-zheng LI Jian-wu
(School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China)

**Abstract** An effective way to deal with massive datasets is to decompose an algorithm into a series of irrelevant tasks, and then to execute them in parallel by using open source softwares. Among overlapping community detection algorithms, the methods based on local expansion in its expansion phase only need the information of local communities and their corresponding neighbors, thus they have the possibility to be executed in parallel. In this paper, we proposed a parallelizable algorithm utilizing local expansion for overlapping community detection, and implemented it by using open source software Spark. The algorithm consists of four phases. Firstly, a group of irrelevant central vertices are selected and their corresponding local networks are used as seeds. Secondly, the algorithm filters the selected seeds by removing those whose vertices are weakly connected. Thirdly, the algorithm adopts a batch expansion strategy to expand seeds, by adding a group of neighboring vertices into the local community or removing a group of vertices from the local community. Finally, similar communities are merged. Experimental results based on artificial networks and real world networks show that our method is both accurate and efficient.

**Keywords** Complex network, Overlapping community detection, Local expansion, Parallelizable algorithm, Spark

## 1 前言

使用复杂网络来表示数据和它们之间的联系是一种简单有效的方式,并且分析复杂网络有助于深入地理解其所对应的系统。因此复杂网络在现实世界中被广泛使用,如科学家及其对应的共同创作关系可以表示为一个科学家协作网[1]。而复杂网络中的一个社区结构[2]由一组连接紧密的结点组成,且它们与社区外部结点连接稀疏。因此社区结构能够反映出系统中的一些功能性的组织,比如社交网络中的社交圈。在复杂网络中发现社区有着广泛的用途。在推荐系统中,社区发现技术可以用来提高推荐的精度[3],还可以解决推荐系统中的冷启动问题[4];标签系统中往往都存在"歧义标签",即一个标签可以表示多种意思,而使用社区发现技术能够定位歧义标签所处的语境,从而帮助系统正确地理解标签[5]。

重叠社区[6]是原始社区的一种很自然的拓展,它允许一个结点同时属于多个不同的社区。重叠社区在复杂网络中很常见,尤其是在社交网络和信息网络中。例如,在一个社交网络中,如果把一个社交圈视作一个社区,那么根据不同的社交关系,一个人往往会属于不同的社区。近年来,已经有很多方法被提出来发现重叠社区,其中包括基于团过滤的算法[6]、边聚类的算法[7]、标签传播的算法[8]和局部拓展的方法[9-13]。基于局部拓展的方法一般都包括选取种子阶段和拓展种子阶段。Local Fitness Maximization(LFM)算法[9]是一种典型的局部拓展方法,它交替执行选取种子阶段和拓展种子阶段,直到所有的结点都被分配了至少一个社区。在选择种子阶段,LFM 随机地选择一个还没有被分配社区的结点作为种子;在拓展阶段,LFM 通过向局部社区中添加其邻居结点或删除局部社区中的结点来最大化这个局部社区的 fitness 函数值,其

到稿日期:2015-08-01 返修日期:2015-09-18 本文受国家自然科学基金项目(61271374)资助。
张忠正 硕士,主要研究方向为数据挖掘、大数据处理;李建武 副教授,主要研究方向为数据挖掘、模式识别、图像处理。

中 fitness 函数是一种能够有效表达社区连接紧密程度的函数。局部拓展的方法在执行拓展步时往往只需要这个局部社区及其邻居结点的信息，因此这种算法具备并行化执行的可能性；而并行算法是应对海量级数据挑战的一种有效的方式。

Google 发表的关于使用 MapReduce 思想处理海量数据的论文[14]引起了学术界和工业界的广泛关注，并且其开源软件 Hadoop[1] 也已经极其完备。但是许多图算法往往都需要迭代执行，而 Hadoop 却不擅长处理这种类型的任务，因为其无法在任务之间有效地共享信息。伯克利大学的 AMPLab 实验室为此设计了 Spark[2] 开源框架，它在硬盘上执行 MapReduce 的性能比 Hadoop 有 10 倍的提升，而在内存中有 100 倍的提升。并且 Spark 提供了非常高层次的编程抽象，使得开发人员实现同一算法所需的代码量只有 Hadoop 的 1/10。Spark 提供的最重要的编程抽象是 Resilient Distributed Dataset(RDD)[15]，RDD 是一个分布在集群中的可并行执行的、容错的、只读的数据集合，它提供了更丰富的编程抽象，而不仅仅是 map 和 reduce 这两种操作。Spark 还提供了另外的抽象，使得我们能够在并行操作中安全有效地共享变量，比如广播变量(Broadcast Variables)。广播变量是一个只读的变量，它会在集群中的每个节点上复制一份，而不是每个任务拷贝一份，因而能够极大地减少在集群中共享变量所需的网络通信时间。

本文提出一种基于局部拓展的并行化方法。该方法由 4 个阶段组成，首先选出一组相互联系不强的中心结点，取每个中心结点及其邻居结点所形成的一组结点作为一个种子；然后通过计算这些种子的 fitness 函数值来过滤掉一部分不好的种子；接着并行拓展这些种子来发现社区，拓展时使用最大化局部社区 fitness 函数的方法；最后通过合并相似社区来形成最终的结果。通过使用 Spark 所提供的 RDD 及广播变量可以很容易地实现所提算法。

本文第 2 节介绍其他相关工作；第 3 节详细阐述本文所提算法；第 4 节为实验分析；最后对本文进行总结。

## 2 相关工作

### 2.1 选取种子的策略

选取种子是局部拓展方法中一个重要的步骤，研究者已经提出了许多不同的策略。最简单的就是 LFM[9] 中提出的随机选择一个还没有被分配社区的结点作为种子。Greedy Clique Expansion(GCE)算法[10]使用最大团作为种子，但是在网络中发现团本身就是一个计算量很大的过程。此外，局部网络分析的方法也被用来发现种子[11]。在有着较大全局聚类系数[16]并且结点度服从长尾分布[17]的网络中，某些局部网络(egonet)有着很好的导电率值，并且这些局部网络被用来作为种子。受文献[11]的启发，文献[12]提出了一种不同的选取种子的策略：首先使用多层次加权核 $k$-means 算法(Graclus 算法)[18]对网络中的结点进行聚类，然后使用一种基于距离的度量来发现每个结点簇的中心，并使用这些中心结点所对应的局部网络作为种子。在 Youtube 公司，$k$ 个点击率最高但是互相之间相关性不强的视频被选为种子[13]。

### 2.2 Fitness 函数

LFM 算法[9]所使用的 fitness 函数是一个很好的局部优化函数，它能刻画出一组结点连接的紧密程度。并且一个社区是由能获得最大 fitness 函数值的一组结点组成，即添加任何其它结点或删除任何已有结点都会使 fitness 函数值减小。子图 $g$ 的 fitness 函数定义如下：

$$f_g=\frac{k_{in}^g}{(k_{in}^g+k_{out}^g)^\alpha} \tag{1}$$

其中，$k_{in}^g$ 表示子图 $g$ 内部的度的总值，也就是子图 $g$ 内部边总数的 2 倍；$k_{out}^g$ 表示子图 $g$ 外部的度的总值，它等于连接子图内结点和子图外结点的边的个数；$\alpha$ 是一个正实数参数，它被用来控制所形成的社区的大小。相应地，对于子图 $g$ 来说一个结点 $A$ 的 fitness 被定义为：包含 $A$ 的子图 $g+\{A\}$ 的 fitness 与不包含 $A$ 的子图 $g-\{A\}$ 的 fitness 的差值，如式(2)所示：

$$f_g^A=f_{g+\{A\}}-f_{g-\{A\}} \tag{2}$$

通过将社区邻居中 fitness 最大且为正值的结点加入社区即可不断地拓展社区，并且这种贪心优化的方法既简单又高效，因为仅仅通过集合操作(集合交、并)就能够将其实现[10]。

### 2.3 并行化社区发现算法

随着数据量的不断增长，许多原有算法已经不能在有效的时间内给出令人满意的解。而此时 MapReduce 编程范式却不断地取得成功，因而许多研究者开始尝试设计 MapReduce形式的社区发现算法，并在开源平台 Hadoop 上执行程序。2009 年，文献[19]成功地将发现最大团的算法[20]设计为 MapReduce 的形式，并在此基础上利用团渗透[6]的方法来发现重叠社区。2010 年，文献[21]将标签传播算法(Label Propagation Algorithm)[22]改写为 MapReduce 的形式，并使用这个算法解决了在海量数据集中无法快速发现其中的连通单元的问题。2013 年，文献[23]提出了一种新的并行化算法 PSCAN，该算法巧妙地将计算结点之间的结构化相似度的过程改写为 MapReduce 的形式，并利用 Hadoop 集群成功地分析了包含 4 千万用户的 14 亿条关注关系的 twitter 网络。

## 3 算法

基于局部拓展的方法有两个核心步骤：选取种子和拓展种子。而在拓展种子的阶段往往只需要局部社区内结点及其邻居结点的相关信息，所以如果能够挑选出一些相对独立的结点作为种子，就可以并行地拓展这些种子。基于以上的思考，提出一个批量并行的局部拓展方法。首先选出一组相关性不强的种子，然后并行使用优化局部社区 fitness 函数的方法来拓展这些种子。具体来说该算法由 4 个阶段组成：1)选

1) Hadoop, http://hadoop.apache.org/

2) Spark, http://spark.apache.org/

择种子;2)过滤种子;3)拓展种子;4)合并高度重叠社区。

### 3.1 选择种子

局部拓展方法的基本思想在于:社区是围绕着几个中心结点形成的。例如社交圈子往往是围绕几个有影响力的人形成的,因此社区的中心结点显然要比边界结点更适合作为种子结点。而社区的中心结点有一个比较明显的特征即它们的度数比较高。在复杂网络中,某些结点及其邻居所组成的局部网络本身就是很好的小社区[11]。由此,提出了一个新的种子选取策略。首先将所有的结点初始化为候选种子集合;然后从中选取一个度数最大的结点 $v$(如果有多个结点的度数相同,则随机选取其中的一个),将结点 $v$ 及其邻居所形成的结点集合作为一个种子,并将 $v$ 及其邻居结点从候选种子集合中删除;重复这个过程直至候选种子集合为空。具体过程如算法 1 所示。

**算法 1** 选择种子

输入:网络 G

输出:种子集合 S,每一个种子包含 3 部分信息:结点集合、内部度数总和 $k_{in}$、外部度数总和 $k_{out}$

1. 将结点按其度数从大到小排序形成一个候选种子列表;
2. 从列表中取出第一个结点 v,将 v 及其邻居结点组成的结点集合作为一个种子加入到 S 中,并初始化该种子:$k_{in}=0$,$k_{out}=0$;
3. 将 v 及其邻居结点从列表中删除;
4. 如果列表不为空,则继续执行步骤 2,否则输出种子集合。

### 3.2 过滤种子

在选取种子的阶段,算法基于的假设是社区的中心结点的度数要比边界结点的高。但是由于网络中也可能存在一些度很高的社区之间的桥接结点(hub),而这个桥接结点及其邻居组成的局部网络显然不是一个好的种子。上述的情况可以通过计算种子的 fitness 函数值来进行判断,因为社区桥接点及其邻居所形成的局部网络的紧密程度要比中心结点及其邻居所形成的局部网络要低。因此如果种子的 fitness 值低于一定程度,则可认为这个种子不是一个好种子并将其删除。具体过程如算法 2 所示。

**算法 2** 过滤种子

输入:网络 G,种子集合 S,控制社区大小的参数 α,fitness 阈值 t

输出:种子集合 S′,其中的每个种子包含 3 部分信息:结点集合、对应的 $k_{in}$ 及 $k_{out}$

1. FOR EACH seed s IN S DO
2. FOR EACH vertex v IN s DO
3. 记 neighbors 为 v 的邻居结点集合;
4. 记 comm 为 neighbors 与 s 中的结点集合的交集;
5. $k_{in}=k_{in}+SIZE(comm)$;
6. $k_{out}=k_{out}+SIZE(neighbors)-SIZE(comm)$;
7. DONE
8. 计算这个种子的 $fitness=\frac{k_{in}}{(k_{in}+k_{out})^{\alpha}}$,若其 fitness 小于 t,则将其删除;
9. DONE
10. 返回处理后的种子集合 S′

### 3.3 拓展

在拓展阶段,算法通过最大化局部社区 fitness 函数的方法拓展社区,它包含两个步骤:添加社区的邻居中 fitness 为正值的结点和删除社区中 fitness 为负值的结点。在拓展阶段,算法采用一种批添加结点和批删除结点的过程。具体来说就是对于一次迭代中的每个社区算法遍历其每一个邻居,记录其 fitness,在遍历结束后统一将 fitness 为正的结点加入社区。然后再遍历一次这个社区,记录其每个结点的 fitness,遍历结束后统一将 fitness 为负的结点移除社区。通过大量的实验发现,迭代执行这两个步骤就能有效地发现社区。

在拓展阶段可以使用文献[10]提到的集合操作来优化计算过程。记录的社区结构包含 3 部分信息:当前社区内的结点集合、当前社区的内部的度 $k_{in}$ 以及当前社区外部的度 $k_{out}$。在计算一个邻居结点的 fitness 时,通过使用集合交操作,计算出加入这个结点之后的 $k'_{in}$ 和 $k'_{out}$,其 fitness 值为 $\frac{k'_{in}}{(k'_{in}+k'_{out})^{\alpha}}-\frac{k_{in}}{(k_{in}+k_{out})^{\alpha}}$。而在计算社区内结点的 fitness 时,通过使用集合交操作,计算出删除该结点之后的 $k'_{in}$ 和 $k'_{out}$,其值为 $\frac{k_{in}}{(k_{in}+k_{out})^{\alpha}}-\frac{k'_{in}}{(k'_{in}+k'_{out})^{\alpha}}$。具体过程如算法 3 和算法 4 所示。

**算法 3** 拓展局部社区

输入:网络 G,一个局部社区 c,控制社区大小的参数 α

输出:拓展之后的社区 c′

1. FOR EACH neighbor vertex v of c DO
2. 记 comm 为 v 的邻居集合与 c 中的结点集合的交集,则添加结点 v 进入社区后的 $k'_{in}=k_{in}+2*SIZE(comm)$,$k'_{out}=k_{out}+SIZE(v$ 的邻居集合$)-2*SIZE(comm)$;
3. v 的 $fitness=\frac{k'_{in}}{(k'_{in}+k'_{out})^{\alpha}}-\frac{k_{in}}{(k_{in}+k_{out})^{\alpha}}$;
4. DONE
5. 将 fitness 为正的邻居结点统一加入到社区中,并在每加入一个结点之后用前述的 $k'_{in}$ 和 $k'_{out}$ 更新社区的 $k_{in}$ 和 $k_{out}$ 信息;
6. FOR EACH vertex v of c
7. 记 comm 为 v 的邻居与 c 中的结点集合的交集,则删除结点 v 之后的 $k'_{in}=k_{in}-2*SIZE(comm)$,$k'_{out}=k_{out}-SIZE(v$ 的邻居集合$)+2*SIZE(comm)$;
8. v 的 $fitness=\frac{k_{in}}{(k_{in}+k_{out})^{\alpha}}-\frac{k'_{in}}{(k'_{in}+k'_{out})^{\alpha}}$;
9. DONE
10. 将 fitness 为负的结点统一移除社区,并在每移除一个结点之后用前述的 $k'_{in}$ 和 $k'_{out}$ 更新社区的 $k_{in}$ 和 $k_{out}$ 信息

**算法 4** 批拓展社区

输入:网络 G,种子集合 S,拓展的迭代次数 iter_time,控制社区大小的参数 α

输出:发现的社区集合 C

1. FOR EACH s in S DO
2. i=0;
3. WHILE i<iter_time
4. DO
5. 使用算法 3 拓展这个社区;
6. i=i+1;
7. DONE
8. 将拓展之后的结果 s′加入 C 中;
9. DONE
10. 返回 C

### 3.4 合并重复社区

在选择种子的过程中，由于种子之间的“距离”并不是很远，因此算法有可能在一个真实的社区中选择了几个不同的种子，而这些种子拓展出的最后结果可能非常相似，故最后需要将这些相似的社区进行合并。两个社区是否应该融合可以用它们之间共同结点在较小的社区中所占的比例来表示，如果其比例大于一定的值，则认为这个小的社区被包含在另一个社区中。具体过程如算法 5 所示。

**算法 5** 合并重复社区

输入：社区集合 C，阈值 t

输出：社区集合 C′

1. FOR EACH c IN C DO
2. IF C′中存在和其相似性大于 t 的社区；
3. THEN 将 c 合并至那个社区；
4. ELSE 将 c 加入 C′；
5. 返回 C′；

## 4 实验

为了下文叙述方便，本文算法被命名为 Batch Local Fitness Maximization (BatchLFM)。实验所使用的 Spark 集群环境由 5 台虚拟机组成，master 结点的配置为 8GB 内存、双核，其余的 worker 结点配置为 2GB 内存、双核。在大量的数据集上测试了 BatchLFM，包括人工生成的网络和现实世界中真实的网络。采用标准化的互信息量(Normalized Mutual Information)[9]来测试算法结果的质量，其值介于 0 和 1 之间。如果发现的社区结构和网络中真实的社区结构是完全相同的，那么它们的 NMI 就是 1；反之，如果完全不同，它们的 NMI 就是 0。即 NMI 的值越高，结果越准确。

### 4.1 在生成的网络中的测试

因为生成的网络均带有内建的社区结构，所以可以很容易地测试算法的准确性。文献[24]提出的 LFR 基准网络近年来被广泛地用于测试社区发现算法，因其结点的度和社区的大小均服从幂律分布[17]，而这正是现实世界中的网络的一个基本特性。使用文献[24]提供的 LFR 生成器来生成网络，具体的参数如下：网络的大小为 10k、50k、150k 和 200k，结点平均的度为 15，结点最大的度为 50，最小社区的大小为 20，最大社区的大小为 50。算法的结果如表 1 所列，从中可以看出算法具有很高的准确性，并且只用少量的迭代次数即可给出有效的结果。

表 1 BatchLFM 在不同规模的 LFR 网络中的 NMI 结果

| 网络\迭代次数 | 0 | 1 | 2 | 5 | 10 |
|---|---|---|---|---|---|
| 10k | 0.6946 | 0.9986 | 0.9997 | 0.9997 | 0.9997 |
| 50k | 0.6985 | 0.9983 | 0.9992 | 0.9992 | 0.9992 |
| 150k | 0.6942 | 0.9985 | 0.9994 | 0.9994 | 0.9994 |
| 200k | 0.6961 | 0.9984 | 0.9992 | 0.9992 | 0.9992 |

### 4.2 在真实的网络中的测试

空手道俱乐部网络[25]和海豚网络[26]是两个经常被用于测试社区发现算法的真实网络，它们的真实社区结构已经被研究者发现。空手道俱乐部网络是一个小型的社交网络，它由 34 个结点组成，并且在俱乐部的教练和管理者发生一次冲突之后分裂为两个不同的社区。海豚网络由 62 个结点组成，描绘了生活在新西兰地区的两个不同的家族的社交行为。

对于空手道俱乐部网络，BatchLFM 算法发现两个社区中有 4 个结点(3、9、10、31)同时属于这两个社区，结果如图 1 所示。

对于海豚网络，BatchLFM 算法成功地发现了 2 个社区，其中有 1 个结点(40)同时属于这两个社区，结果如图 2 所示。

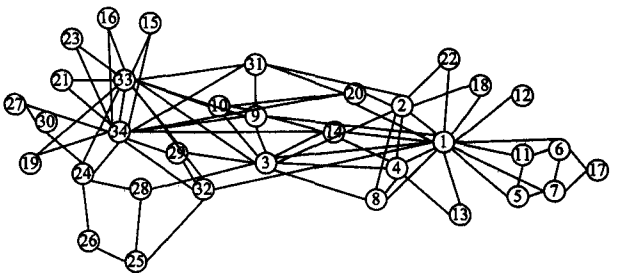

图 1 空手道俱乐部网络的结果

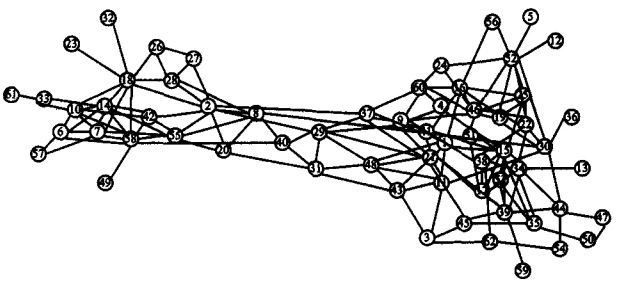

图 2 海豚网络的结果

所提算法在这两个网络上得到的结果与它们对应的真实社区结构几乎相同。实验结果的 NMI 值如表 2 所列。使用 CFinder[1)] 和 Copra[2)] 作为结果的对比，CFinder 是团过滤算法[6]作者提供的软件，而 Copra 算法[8]是一种可发现重叠社区的标签传播算法。

表 2 在空手道俱乐部网络和海豚网络上 3 种不同算法的结果

| 算法\网络 | 空手道俱乐部 | 海豚 |
|---|---|---|
| BatchLFM | 0.743 | 0.944 |
| CFinder | 0.216 | 0.195 |
| Copra | 0.458 | 0.574 |

DBLP 网络[27]、Amazon 网络[27]以及 Youtube 网络[27]是 3 个规模比较大的真实网络，其中 DBLP 网络是一个科学家协作网络，如果两个科学家共同撰写了至少一篇论文，那么他们之间就存在一条边；Amazon 网络是一个商品被共同购买的网络，如果两个商品经常被客户同时购买，那么它们之间存在一条边；Youtube 网络是由用户在 Youtube 网站上的社交关系形成的。这 3 个数据集的大小以及 BatchLFM 算法在其上运行所需的时间如表 3 所列。

表 3 网络的规模及运行时间

| 网络 | 结点数量 | 边数量 | 运行时间(min) |
|---|---|---|---|
| DBLP | 317080 | 1049866 | 1.4 |
| Amazon | 334863 | 925872 | 1.8 |
| Youtube | 1134890 | 298724 | 8.9 |

1) Palla G, Derenyi I, Farkas I, et al. The software of Clique Percolation Method [CP/OL]. 2005[2012-09-10], http://www.cfinder.org

2) http://www.cs.bris.ac.uk/~steve/networks/copra

**结束语** 本文提出了一种可并行执行的批量式的局部拓展方法 BatchLFM。它的主要思想就是一次性地选出一组互相之间关联性不强的种子，然后通过优化 fitness 函数的方法拓展种子。通过使用 Spark 所提供的高层次的编程抽象 RDD 和广播变量可以轻易地实现本文描述的算法，并且在执行过程中使用集合操作能够极大地提升算法的性能。在广泛的数据集(包括人工生成的 LFR 网络及多个真实世界中的网络)上的实验证明了 BatchLFM 的有效性及高效性。

## 参 考 文 献


[1] Newman M E J. The structure of scientific collaboration networks[J]. Proceedings of the National Academy of Sciences, 2001,98(2):404-409

[2] Girvan M, Newman M E J. Community structure in social and biological networks[J]. Proceedings of the National Academy of Sciences,2002,99(12):7821-7826

[3] Deng K, Zhang J, Yang J. Mobile Recommendation Based on Link Community Detection[J]. Scientific World Journal, 2013, 2014(11):259156

[4] Sahebi S,Cohen W W. Community-based recommendations:a solution to the cold start problem[C]// Workshop on Recommender Systems and the Social Web. RSWEB,2011

[5] Au Yeung C, Gibbins N, Shadbolt N. Contextualising tags in collaborative tagging systems[C]// Proceedings of the 20th ACM Conference on Hypertext and Hypermedia. ACM, 2009: 251-260

[6] Palla G, Derényi I, Farkas I, et al. Uncovering the overlapping community structure of complex networks in nature and society [J]. Nature,2005,435(7043):814-818

[7] Ahn Y Y, Bagrow J P, Lehmann S. Link communities reveal multiscale complexity in networks [J]. Nature,2010,466 (7307):761-764

[8] Gregory S. Finding overlapping communities in networks by label propagation[J]. New Journal of Physics,2009,12(10):2011-2024

[9] Lancichinetti A, Fortunato S, Kertész J. Detecting the overlapping and hierarchical community structure in complex networks [J]. New Journal of Physics,2009,11(3):033015

[10] Lee C, Reid F, McDaid A, et al. Detecting highly overlapping community structure by greedy clique expansion[C]// International Conference on Paper Presented at Sna-kdd Workshop. 2010

[11] Gleich D F, Seshadhri C. Vertex neighborhoods, low conductance cuts, and good seeds for local community methods[C]// Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM,2012:597-605

[12] Whang J J, Gleich D F, Dhillon I S. Overlapping community detection using seed set expansion[C]// Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management. ACM,2013:2099-2108

[13] Gargi U, Lu W, Mirrokni V S, et al. Large-Scale Community Detection on YouTube for Topic Discovery and Exploration[C]// ICWSM. 2011

[14] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113

[15] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing[C]// Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. USENIX Association,2012:2-2

[16] Watts D J, Strogatz S H. Collective dynamics of 'small-world' networks[J]. Nature,1998,393(6684):440-442

[17] Barabási A L, Albert R. Emergence of scaling in random networks[J]. Science,1999,286(5439):509-512

[18] Dhillon I S, Guan Y, Kulis B. Weighted graph cuts without eigenvectors a multilevel approach[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29(11): 1944-1957

[19] Yang S, Wang B, Zhao H, et al. Efficient dense structure mining using MapReduce[C]// IEEE International Conference on Data Mining Workshops, 2009(ICDMW'09). IEEE,2009:332-337

[20] Tomita E, Tanaka A, Takahashi H. The worst-case time complexity for generating all maximal cliques and computational experiments[J]. Theoretical Computer Science, 2006, 363(1): 28-42

[21] Wu B, Du Y H. Cloud-based connected component algorithm[C]// 2010 International Conference on Artificial Intelligence and Computational Intelligence (AICI). IEEE,2010,3:122-126

[22] Raghavan U N, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks[J]. Physical Review E,2007,76(3):036106

[23] Zhao W, Martha V, Xu X. Pscan: A parallel structural clustering algorithm for big networks in mapreduce[C]// 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA). IEEE,2013:862-869

[24] Lancichinetti A, Fortunato S. Community detection algorithms: a comparative analysis[J]. Physical review E,2009,80(5):056117

[25] Zachary W W. An information flow model for conflict and fission in small groups[J]. Journal of Anthropological Research, 1977, 33(4):452-473

[26] Lusseau D, Schneider K, Boisseau O J, et al. The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations[J]. Behavioral Ecology and Sociobiology,2003,54(4):396-405

[27] Yang J, Leskovec J. Defining and Evaluating Network Communities Based on Ground-Truth[C]// 2012 IEEE 12th International Conference on Data Mining. IEEE Computer Society, 2012:745-754