大规模数据下的社交网络结构洞节点发现算法研究

王 珍[1] 韩忠明[2] 李 晋[3]
(电子工程学院网络系 合肥 230037)[1] (北京工商大学计算机与信息工程学院 北京 100048)[2]
(北京信息科技大学 北京 100101)[3]

摘 要 随着社会网络数据规模的递增,结构洞节点计算涉及的计算量呈几何级增长,如何构建有效的并行化算法并缩短算法运行的时间成为当前研究的难点。针对大规模数据量下结构洞节点发现算法的不足,利用并行化思想设计实现了基于MapReduce的结构洞节点发现算法。该算法通过DBLP,YouTube和Califonia公路网这3组规模不同的数据集在Hadoop集群上运行的实验结果表明,增加DataNode机器节点的数量能够缩短算法运行的时间,提高运行效率且具有良好的并行加速比和扩展性能。

关键词 并行计算,社会网络,结构洞,节点发现
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.041

Research on Social Network Structural Holes Discovery Algorithm under Large-scale Data

WANG Zhen[1] HAN Zhong-ming[2] LI Jin[3]
(Department of Network, Electronic Engineering Institute, Hefei 230037, China)[1]
(College of Computer and Information Engineering, Beijing Technology and Business University, Beijing 100048, China)[2]
(Beijing Information Science and Technology University, Beijing 100101, China)[3]

Abstract With the increase of social network data scale, the amount of calculation on structural holes exponentially increases. How to construct efficient parallel algorithms to shorten the running time of algorithms becomes the difficulties of the current study. This paper concentrated on shortages of network structural holes discovery algorithm, using parallel thought to design structural holes discovery algorithm based on MapReduce. The algorithm uses three groups of different sizes of data sets which are DBLP, YouTube and California road network to test on Hadoop. The results show that increasing the number of Data Node's machine nodes can shorten the running time of algorithms and increase operational efficiency, and this algorithm has good parallel speedup and scalability.

Keywords Parallel computing, Social networks, Structural holes, Node discovery

社交新媒体的引入极大地丰富了人们的生活,人与人之间可以通过网络文字、媒体的交互来进行即时通信,这种基于博客、QQ、微博等社交媒体所形成的人与人之间的协作关系网络称为社交网络。网络中每个人都可以作为一个节点参与到信息的传播和扩散中,由于每个人的属性、发布消息的多少、个人影响力等因素导致该节点在网络拓扑中的地位不同,如何发现和识别关键节点是研究社交网络信息传播的关键,且对于控制恶意信息传播、控制谣言蔓延都有理论价值。

针对关键节点的发现识别已经展开了相应的研究,如节点度值、紧密中心性、介数中心性、PageRank、HITS等,这些研究都是从网络结构特征出发对节点进行衡量[1]。文献[2]认为社团结构在信息传播过程中对信息的扩散具有抑制作用,网络社团之间的联络强度是制约网络信息传播的瓶颈。结构洞的节点是从网络集群拓扑中衍生出来的一种节点重要程度的度量方法,起源于社会学中的竞争团体的理论,最早由Burt提出[7]。从社会学角度来看,结构洞是指两组人群之间联络的"瓶颈";从复杂网络的拓扑视角来看,结构洞节点的存在是信息传播的基石,研究社交网络结构洞节点发现算法有利于分析信息传播的机理。对于结构洞的研究,文献[3]基于结构洞来衡量社交网络中团体的互动竞争关系;文献[4]根据节点邻居列表和相邻节点的拓扑结构提出了一种局部测量中心性的算法,在邻居结构的基础上综合节点的网络约束系数来衡量结构洞节点的"桥接"强度;文献[5]通过刻画内部节点的协同作用来识别信息传播过程中的结构洞节点,将内部节点重要度和社团之间的重要度结合,并针对新浪微博的数据验证了算法的有效性;文献[6]针对社会关系网络的演化机理进行研究,利用结构洞的判定来推测隐结构。结构洞节点发现算法与以度值中心性为代表的算法不同,在计算过程中复杂度较高,尤其是当社交网络中节点、链路、文本的数量规模激增时,如何保证发现算法的性能和效率是面临的主要难题。MapReduce[10]作为一种并行化的优化算法框架,可以集成普通、价值低廉的计算设备,在不降低可靠性和容错性的基础上

到稿日期:2015-11-15 返修日期:2016-03-17 本文受国家自然科学基金项目(61170112),北京市教委科研计划项目(SM201411232005)资助。
王 珍(1981—),女,博士,工程师,主要研究方向为复杂网络、社会网络分析,E-mail:405573422@qq.com;韩忠明(1972—),男,副教授,主要研究方向为大数据分析与挖掘;李 晋(1977—),女,硕士,讲师,主要研究方向为网络与新媒体传播,E-mail:bjlijin@126.com。

处理 T 级别的数据集,将其应用到社交网络结构洞节点发现算法中以缩短算法运行时间,有助于挖掘社交网络中有价值的信息。因此,本文针对大规模数据量下结构洞节点发现算法的不足,利用并行化思想设计实现了基于 MapReduce 的结构洞节点发现算法。

本文首先介绍了结构洞节点发现中的 7 个度量指标,分别是网络约束系数(CT)、网络有效规模(ES)、效率(EF)、等级度(HI)、局部聚类系数(CC)、介数中心性(BT);然后设计数据结构和结构洞节点并行化发现算法框架,并对 Map 函数、Reduce 函数以及 Main 函数进行并行化实现;最后针对效率、并行加速比和扩展性 3 个评价标准进行了仿真验证。

1　结构洞节点

如图 1 所示,群体 A 和群体 B 两组人既交织在一起但又泾渭分明,若 B 组人需与 A 组中冗余的人 b 通信则必须经过节点 a。其中节点 a 起着中间人的过渡作用,若移除该节点则网络将分裂成两个不连通的子分支。

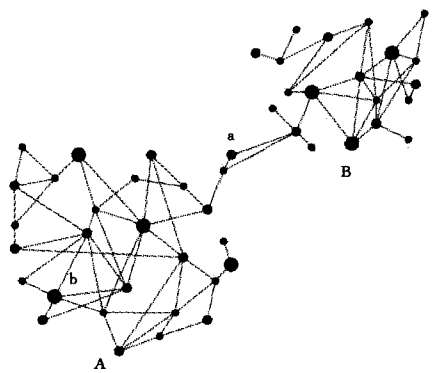

图 1　结构洞节点示意图

Burt 提出的结构洞的度量方法如表 1 所列。

表 1　结构洞度量指标

	度量指标	计算方法
CT	网络约束系数	$C_{ij}=(P_{ij}+\sum_{q}P_{iq}P_{qj})^2$
ES	网络有效规模	$ES_i=n-\sum_{i}\sum_{j}P_{ij}$
EF	效率	$EF_i=ES_i/n$
HI	等级度	$EF_i=ES_i/n$
CC	局部聚类系数	$CC(i)=\frac{2E(i)}{k(i)[k(i)-1]}$
BT	介数中心性	$BT(q)=\sum_{i}\sum_{j}g_{ij}(q)/g_{ij}$
PR	Pagerank	$PR(P_i)=\frac{1-q}{N}-q\sum_{P_j}\frac{PR(P_i)}{L(P_i)}$

表 1 中 P_{ij} 表示节点 i 的所有邻居列表中节点 j 的占比;g_{ij} 表示节点 i 和节点 j 之间的最短路径经过节点 q 的条数;$L(P_j)$ 是节点 P_j 的邻居节点数量,相应的详细指标描述见文献[8-9]。下面将在大规模数据下对结构洞节点发现算法进行并行化设计。

2　大规模数据下的并行化设计

2.1　数据结构设计

社会网络节点数据即为节点间的关系,可以抽象成图中的节点,即采用 MapReduce 框架对图进行处理,其中的重要问题即为 Map 函数和 Reduce 函数对图中节点和边数据的处理。由于 MapReduce 以数据对的形式定义数据格式(如{key,value}),因此以数据对的形式定义图中的节点数据结构(如{id1,id2})。其中 key 即为节点 id,通过节点 id 对MapReduce的输入数据进行分割,将输入数据映射到不同的 Map 工作节点上,即上节提到的 JobTracker 中,随后将 Map 阶段得出的输出以同样的数据结构映射到不同的 Reduce 工作节点上,其中都是以 key 值也即节点 id 作为映射的依据,而 value则为对应的节点对象,节点对象是该节点连接的所有节点,记录节点与图中其他节点的关系信息。

MapReduce 根据节点 id 对输入数据进行切分,每一个切分中的 key 值即为节点 id,value 值为其节点对象信息。MapReduce对数据进行切分后,数据被分为多份,切分后的数据通过节点 id 的映射被送往不同的 JobTracker 进行处理。Map 节点的输出形式为{key, value}形式,Map 阶段的输出作为 Reduce 阶段的输入,一般情况下,多个 Map 的输出会成为一个 Reduce 的输入,所以 Map 阶段的输出有可能需要经过一定的合并排序处理后再输入到 Reduce 阶段,即 Map 阶段的{id1,id2}数据对的数目要小于 Reduce 阶段,数据中的节点 id 即为重要的映射依据。

2.2　并行化设计

MapReduce 并行计算的优势是利用 Map 和 Reduce 两个阶段在普通机器集群上实现大规模数据集的并行计算。通过分析构建结构洞节点重要性度量模型的过程可知,结构洞节点并行化发现算法的计算量集中在两个方面:1)需要计算社会网络中节点的 7 个面向结构洞的节点重要性度量指标(见表 1);2)通过基于 ListNet 的结构洞节点重要性度量模型来学习参数值的过程。下节利用 MapReduce 框架对这两方面进行最大化并行计算,以此来解决在数据量较大时该算法的局限性问题。

由图 2 知,基于 MapReduce 的结构洞节点发现算法框架的设计如下。

(1)对输入的社会网络节点原始数据 data. txt 进行数据处理,原始数据的数据格式为边的端点对,将该数据进行MapReduce程序处理后输出 key/value 键值对的形式,数据结构为{key=id,value=[neighbors]},保存的结果文档为 link. txt 且上传到 HDFS 中,以待使用。

(2)并行化计算面向结构洞的节点重要性度量指标值。输入文件 link. txt,将面向结构洞的节点重要性度量指标算法改写为 MapReduce 程序,利用 Map 函数和 Reduce 函数并行计算出每个节点的 7 个度量指标值,输出节点与对应指标值文档 index. txt 且上传到 HDFS 中。

(3)采用并行化的基于 ListNet 的结构洞节点重要性度量模型算法学习模型参数。输入文档 index. txt、记录参数的文档 parameter. txt 以及判断文档 y. txt,最初上传到 HDFS 中的 parameter. txt 文档中有参数的初始值,y. txt 文档保存的是节点相关性判断条件,调用 indexs. txt 中保存的数据来计算各节点的得分函数、节点的概率值 P 以及损失函数,学习每个节点每个参数的更新量;累计叠加步骤(2)中每个参数

的更新量，并计算参数更新量的平均值，最后输出这个平均值，更新参数，将输出保存到参数文档 parameter. txt。

基于 MapReduce 的结构洞节点排序算法的实现步骤为：

(1)输入训练数据集，数据集格式为(key，value)对，即 key 为社会网络节点，value 为节点对应的值；

(2)执行多个 Map 任务，每个 Map 任务计算每个节点每个参数的更新量 $\Delta\theta$；

(3)执行 Reduce 任务，累计叠加每个参数的更新量，并计算参数更新量的平均值，最后输出该平均值；

(4)更新参数值；

(5)循环执行步骤(2)—步骤(4)，直到参数不再发生明显变化。

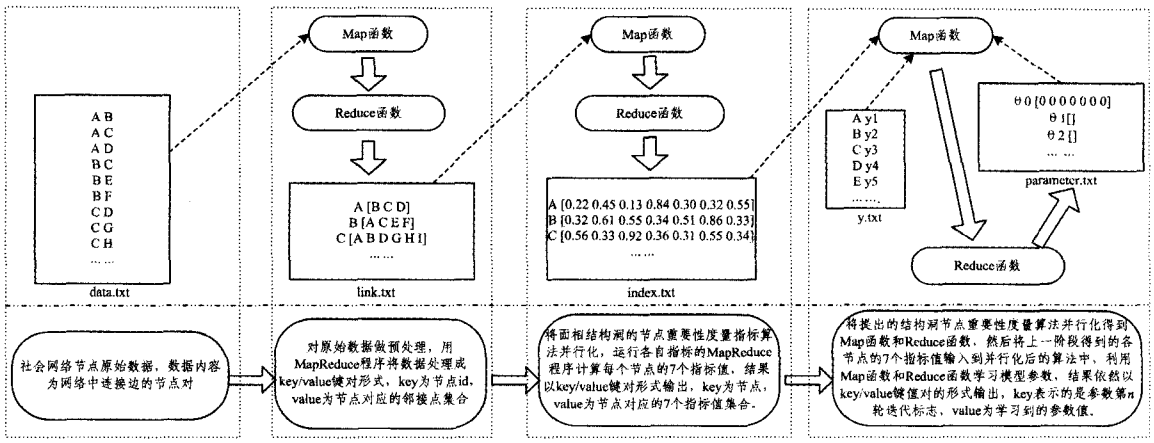

图 2 结构洞节点并行化发现算法框架图

3 基于 MapReduce 的关键节点发现算法

在第 2 节的并行算法设计中，Map 阶段和 Reduce 阶段是核心部分，且会在限制条件下迭代计算，下面详细介绍 Map 函数、Reduce 函数以及 Main 函数的计算过程。

3.1 Map 阶段

Map 阶段输入的数据对{key=id，value=[BT，CT，EF，ES，CC，PR，HT，y]}中 key 是社会网络中的每个节点的 id，value 包括度量指标 CT，BT，PR，CC，ES，EF，HI 和相关性判断条件 y；Map 阶段的输出结果是每个参数 θ 和对应的参数更新量 $\Delta\theta$。Map 函数的基本过程是首先计算社会网络中每个节点的评分函数 $h(x_i)$，该评分函数中带有需要学习的参数 θ，根据评分函数可得出排列概率：

$$P_{h(x)}=\prod_{j=1}^{q}\frac{\exp(h(x_j))}{\sum_{t=k}^{n}\exp(h(x_t))} \tag{1}$$

损失函数为：

$$L(y,h_\theta(x))=-\sum_{i=1}^{n}P_y * \log(P_{h_\theta(x)}) \tag{2}$$

根据损失函数计算参数更新量 $\Delta\theta$，记录结果为数据对{key=θ，value=$\Delta\theta$}，并将该结果输入到 Reduce 阶段中。Map 函数伪代码如下所示。

Input：{key=id，value=[BT，CT，EF，ES，CC，PR，HT，y]}

Output：{θ，$\Delta\theta$}

1. Input{(key_1，$value_1$)，(key_2，$value_2$)，…，(key_m，$value_m$)}
2. for i=1 to m do：
3. Compute $h_\theta(x_i)$ with current $\vec{\theta}$
4. Compute $P_{h(x)}$ with $h_\theta(x_i)$ in the network
5. Get $P_{h(x)}$ into neural network and compute $L(y,h_\theta(x))$ with y_i
6. Compute $\Delta\vec{\theta}$
7. Return {θ，$\Delta\theta$}

该算法是基于介数中心性设计的，表 1 中其他 6 类度量指标的计算需要在不同计算机上调用不同的函数来分别进行，在 Map 阶段完成原算法，Reduce 阶段进行结果合并输出即可。由此通过调用不同的函数并行计算即可得到 7 类指标来最终实现合并输出，由于不同指标的数学计算公式不同，其函数的编写也不同。

3.2 Reduce 阶段

将 Map 阶段输出的数据对{key=θ，value=$\Delta\theta$}作为 Reduce 阶段的输入，输出结果为数据对{θ，θ'}，即每个参数 θ 与对应参数的更新值 θ'。Reduce 函数的基本过程是将 Map 阶段的结果进行合并且累计叠加，计算出每个参数的平均更新量 avg($\Delta\theta$)，然后根据公式：

$$\theta'=\theta-\eta\cdot \text{avg}(\Delta\theta) \tag{3}$$

计算新的参数值。Reduce 函数伪代码如下所示。

Input：{key=θ，value=$\Delta\theta$}

Output：{θ，θ'}

1. Input {key=θ，value=$\Delta\theta$}
2. sum=0
3. counter=0
4. while(value is not empty)
5. sum +=value2
6. counter +=1
7. avg $\Delta\theta$=sum/counter
8. Update $\theta'=\theta-\eta\cdot \text{avg}(\Delta\theta)$
9. Return {θ，θ'}

3.3 Main 函数

Main 函数控制整个作业的迭代次数，由于所提算法是一个迭代求解的过程，因此需要迭代运行作业，输入初始的参数值 θ，运行 Map 函数，再通过 Reduce 函数得出参数的更新值 θ'，然后判断参数值变化的大小，若参数误差较大，则返回步骤 2 和步骤 3 以继续进行迭代计算，当参数误差降到可接受范围内，迭代结束。Main 函数的伪代码下所示。

1. Input current θ
2. Map(key1，value1)
3. Reduce(key2，value2)

4. if $\theta'-\theta>\varepsilon$:
5.　　go back to the step 2 and step 3
6. else:
7.　　End

3.4 性能评估

检验并行化算法性能的主要标准有 3 种：效率、并行加速比和扩展性，详细信息介绍如下。

(1)效率：在数据规模不断变化时完成并行化计算任务所需要消耗的时间，表示为 Ef。

(2)并行加速比：在一个节点设备上完成任务所需要消耗的时间 T_s 与在多个节点设备上完成任务所消耗的时间 T_k 之比，表示为 $Speedup$，计算公式如下。

$$Speedup=\frac{T_s}{T_k} \tag{4}$$

(3)扩展性：数据规模与节点设备数量同增长的情况下，DataNode 机器节点的执行效率，表示为 $Scaleup$。

4 实验分析

4.1 Hadoop 集群实验环境

在 Hadoop 平台上进行实验，选用 Hadoop 的 0.20.2 版本，采用该版本的配置文件 core-site.xml 配置 Hadoop 的 Namenode 的 IP 地址和端口，采用 hdfs-site.xml 配置 HDFS 的数据存放地址以及数据块复制数量，采用 mapred-site.xml 配置 JobTracker 的地址和端口。

Hadoop 集群的硬件环境为 7 台 x86 架构的 PC 机，每个节点设备都采用 Intel(R) 主频为 2.2GHz 的 CPU，内存为 2GB，硬盘容量为 160GB，节点之间通过 Gigabit Ethernet 交换机连接，将其中一台机器作为 NameNode 和 JobTracker，其他机器作为 DataNode 和 TaskTracker，各节点设备操作系统环境为 CentOS Linux。

4.2 数据采集与描述

实验采用 3 组数据集作为测试数据：DBLP 数据集、YouTube 数据集和 California 公路网数据集。本文采用这 3 组数据集验证算法性能，3 组数据集规模依次递增，如表 2 所列。

表 2　实验数据集

网络	节点数量	边数量	数据集大小/MB
DBLP	317080	1049866	13.2
YouTube	1134890	2987624	36.9
California	1965206	2766607	83.7

DBLP 数据集是计算机科学领域中一组发表研究论文的作者之间的关系网数据，本文将该数据集中的每个作者作为社会网络中的节点，至少共同发表过一篇论文的两个作者会产生关联，这种关联即为社会网络中节点之间的链接，该数据集有 31 万多个节点，边的数量也达到了 105 万条，符合本实验数据量较大的特点。

YouTube 数据集是 YouTube 社交网站注册用户关系数据集，YouTube 是一个视频共享网站，也拥有其社交网络，注册的用户在社交网络中相互成为好友，这个真实的社交网络适用于本实验。

California 公路网数据集是加利福尼亚州的一个公路网络，公路中的十字路口和端点表示节点，公路则为边，组成了一个无向社会网络。该数据集是本文采用的规模最大的数据集，网络中节点数量达到了 196 万个，边的数量也达到了 276 万条，可以通过这个数据集来检验本文并行化算法的性能。

4.3 并行算法验证

在算法的并行化过程中，对每一个 txt 文件进行操作的过程均要经过 MapReduce 函数。首先基于 MapReduce 的面向结构洞节点重要性度量指标算法，将这部分算法结果作为下一部分算法的输入，即基于 MapReduce 的结构洞节点重要性度量算法，这两个过程承上启下，息息相关。但是核心节点的度量公式和结构洞算法是一致的，即是否采用并行算法对结构洞算法的准确性并无影响。因此，在并行算法验证过程中只对效率、并行加速比和扩展性这 3 个性能指标进行验证。将 3 组数据集进行一定的处理后存储到 Hadoop 集群的 HDFS 中，在 Hadoop 集群中指定其中一台机器作为 NameNode 和 JobTracker，指定另一台机器作为 SecondaryNode 来备份元数据，以在 NameNode 出现意外宕机时能够有效地保存元数据且保证实验的正常运行，其他 5 台都作为 DataNode 和 TaskTracker。启动 MapReduce 程序运行算法，训练出的参数结果如表 3 所列。该实验结果表明算法能够收敛，运行稳定，而且得出的数据结果满足实际情况，这初步证明了该算法能够适用于数据量较大的社会网络。

表 3　数据集测试结果

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7
DBLP	−1.48	4.20	−0.24	−1.24	−1.03	−0.24	0.43
YouTube	−1.47	3.88	−0.22	−1.12	−0.82	−0.18	0.31
California	−1.68	4.21	−0.28	−1.34	−1.13	−0.26	0.46

为了进一步验证在不同数量的 DataNode 机器节点的情况下数据规模的改变对所提算法性能的影响，采用 DBLP，YouTube 和 Califonia 这 3 个规模逐渐递增的数据集在 1～5 台 DataNode 机器节点上进行实验，从效率、并行加速比和扩展性 3 个方面来评价基于 MapReduce 的结构洞节点算法。

图 3 示出了 3 个数据集在 DataNode 机器节点逐渐增加的情况下算法运行所耗费的时间趋势折线图，本文将从此角度来分析算法的效率性能。从图中可以看出，3 个规模的数据集在 DataNode 数量增加的情况下运行时间变短，说明并行化算法在机器节点增多的情况下任务被分配出去，减少了 DataNode 机器节点执行的任务量，同时也节省了运行时间。从 3 个数据集运行时间的趋势的对比来看，数据规模较大的 California 数据集在 DataNode 节点数量增大时运行时间减少的幅度比其他两个数据集大，DBLP 和 YouTube 两个数据集在 DataNode 数量为 4 和 5 时运行时间变化较小。该对比说明当数据量较大时分配的 DataNode 的个数越多，Map 任务的分片越多，能够有效地缩短算法程序运行的时间。

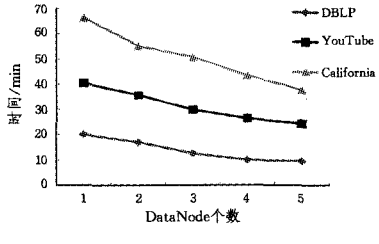

图 3　Ef 性能对比图

图 4 是 3 个数据集在不同数量的 DataNode 机器节点下

运行并行化算法的加速比，其很直观地展示出了在 DataNode 机器节点数增加的情况下并行加速比也随之增大。与此同时，图中结果也表明数据规模越大，并行加速比越大。结果表明，本文并行化算法具有优良的加速比，虽然在数据量和 DataNode 机器节点增加时，Map 函数的输出结果与 Reduce 函数执行通信时的负荷会增加，但是由于本文算法的数据结构设计得合理，因此在通信过程中耗时不明显，从而使得在数据量和 DataNode 机器节点数量同时增加的情况下算法的并行加速比的性能增强。从图中还可以看出，数据集 DBLP 和 YouTube 在 DataNode 机器节点数量增加时 Speedup 的增长趋势比较平稳，而数据集 California 数据集在 DataNode 机器节点从 4 个变为 5 个时 Speedup 增长量比之前小，这也说明在数据量较大时，增加 DataNode 机器节点会增加通信负荷，也会在一定程度上影响执行效果，所以并不是在任何情况下无限制地增加 DataNode 机器节点数量都会产生好的效果。

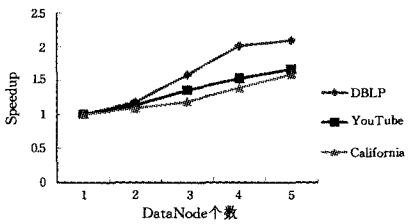

图 4 Speedup 性能对比图

为验证所提算法的扩展性，将不同规模的数据集对应不同数量的 DataNode 节点，DBLP 数据集对应 1 个 DataNode 机器节点，YouTube 数据集对应 3 个 DataNode 机器节点，Califonia 公路网数据集对应 5 个 DataNode 机器节点。从图 5 中可以看出，当 DataNode 数目和数据规模同时增大时，算法的 Scaleup 值减小，说明数据规模变大时每一个 DataNode 机器节点发挥出更大的计算优势，即算法的 Scaleup 性能随着数据量增大而变得更好。从 Scaleup 的变化趋势可以看出，当 DataNode 机器节点数变大时，Scaleup 值开始为 1，而后逐步小于 1，并且越来越小，证明本文并行性算法对数据集规模的变化有很好的适应性。

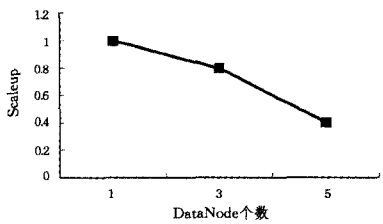

图 5 Scaleup 性能对比图

结束语 为解决提出的基于结构洞节点发现算法在面对数据量较大时表现的不足，将该算法并行化，同时设计并实现了基于 MapReduce 的结构洞节点发现算法。算法采用 DBLP，YouTube 和 Califonia 公路网这 3 组规模不同的数据集在 Hadoop 集群上进行实验，用效率、并行加速比和扩展性 3 个评价标准来衡量基于 MapReduce 的结构洞节点发现算法的性能，实验结果表明所提算法在数据规模增大的情况下，增加 DataNode 机器节点的数量能够缩短算法运行的时间，即其效率得到了提高。与此同时，实验结论也说明了本文并行化算法具有良好的并行加速比和扩展性能。

参 考 文 献

[1] YUAN W G, LIU Y, CHENG J J, et al. Empirical analysis of microblog centrality and spread influence based on Bi-directional connection[J]. Acta Physica Sinica, 2013, 62(3): 494-503. (in Chinese)
苑卫国，刘云，程军军，等. 微博双向“关注”网络节点中心性及传播影响力的分析 [J]. 物理学报，2013，62(3)：494-503.

[2] LUO Z G, DING F, JIANG X Z, et al. New Progress on Community Detection in Complex Networks[J]. Journal of National University of Defense Technology, 2011, 33(1): 47-52. (in Chinese)
骆志刚，丁凡，蒋晓舟，等. 复杂网络社团发现算法研究新进展 [J]. 国防科技大学学报，2011，33(1)：47-52.

[3] DI VINCENZO F, HEMPHÄLÄ J, MAGNUSSON M, et al. Exploring the role of structural holes in learning: an empirical study of Swedish pharmacies[J]. Journal of Knowledge Management, 2012, 16(4): 576-591.

[4] XIA S, DAI B T, LIM E P, et al. Link prediction for bipartite social networks: the Role of Structural Holes[C]// 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE, 2012: 153-157.

[5] SU X P, SONG Y R. Leveraging neighborhood “structural holes” to identifying key spreaders in so cial networks[J]. Acta Physica Sinica, 2015, 64(2): 1-11. (in Chinese)
苏晓萍，宋玉蓉. 利用邻域“结构洞”寻找社会网络中最具影响力节点[J]. 物理学报，2015，64(2)：1-11.

[6] YANG J Z. Research on the Identifying Nodes in Online Social Network [D]. Taiyuan: Taiyuan University of Technology, 2014. (in Chinese)
杨敬宗. 在线社会网络影响力节点发现方法研究[D]. 太原：太原理工大学，2014.

[7] WANG L, CHENG S Q, SHEN H W, et al. Structure Inference and Prediction in the Co-Evolution of Social Networks[J]. Journal of Computer Research and Development, 2015, 50(12): 2492-2503. (in Chinese)
王莉，程苏琦，沈华伟，等. 在线社会网络共演化的结构推断与预测[J]. 计算机研究与发展，2015，50(12)：2492-2503.

[8] BURT R S, KILDUFF M, TASSELLI S. Social Network Analysis: Foundations and Frontiers on Advantage[J]. Annual Review of Psychology, 2013, 64(2): 527-547.

[9] YAN H L, XIANG J U, ZHANG X Y, et al. Community detection using global and local structural information[J]. Pramana, 2013, 80(1): 173-185.

[10] HAN Z M, WU Y, TAN X S, et al. Comparison and analysis on measure indexes for structural hole nodes in social network[J]. Journal of Shandong University (Engineering Science), 2015, 1(1): 1-8. (in Chinese)
韩忠明，吴杨，谭旭升，等. 社会网络结构洞节点度量指标比较与分析[J]. 山东大学学报(工学版)，2015，1(1)：1-8.

[11] SHI Q, XIAO Y H, WEN W H, et al. Research on method for extracting large-scale social network based on Mapreduce[J]. Application Research of Computers, 2011, 1(1): 145-148. (in Chinese)
施佺，肖仰华，温文灏，等. 基于 Mapreduce 的大规模社会网络提取方法研究[J]. 计算机应用研究，2011，1(1)：145-148.