在线社交网络个体影响力算法测试与性能评估

全拥[1]，贾焰[1]，张良[1]，朱争[1]，周斌[1]，方滨兴[2]
（1. 国防科技大学计算机学院，湖南 长沙 410073；2. 北京邮电大学计算机学院，北京 100876）

摘 要：社交影响力是驱动信息传播的关键因素，基于在线社交网络数据，可以对社交影响力进行建模和分析。针对一种经典的个体影响力计算方法，介绍了该算法的 2 种并行化实现，并在真实大规模在线社交网络数据集上进行了性能测试。结果表明，借助现有的大数据处理框架，显著提高了个体影响力计算方法在海量数据集中的计算效率，同时也给该类算法的研究和优化提供了实证依据。

关键词：性能测试；社交影响力；分布式计算；在线社交网络
中图分类号：TP391
文献标识码：A
doi: 10.11959/j.issn.1000−436x.2018217

Performance analysis and testing of personal influence algorithm in online social networks

QUAN Yong[1], JIA Yan[1], ZHANG Liang[1], ZHU Zheng[1], ZHOU Bin[1], FANG Binxing[2]
1. College of Computer, National University of Defense Technology, Changsha 410073, China
2. College of Computer, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract: Social influence is the key factor to drive information propagation in online social networks and can be modeled and analyzed with social networking data. As a kind of classical personal influence algorithm, two parallel implementation versions of a PageRank based method were introduced. Furthermore, extensive experiments were conducted on a large-scale real dataset to test the performance of these parallel methods in a distributed environment. The results demonstrate that the computational efficiency of the personal influence algorithm can be improved significantly in massive data sets by virtue of existing big data processing framework, and provide an empirical reference for the future research and optimization of the algorithm as well.

Key words: performance testing, social influence, distributed computing, online social networks

1 引言

随着 Web 2.0 技术的进一步完善以及移动智能终端的大量使用，在线社交网络蓬勃发展。以新浪微博和 Facebook 为代表的在线社交网络平台逐渐成为网络应用的主流，并改变了人们生活和交流的方式。在线社交网络中用户之间的交互行为，使网络世界与现实世界相互影响，特别是快速传播扩散的网络信息能够迅速形成社会舆论，对现实世界人们的行为产生直接影响[1]。社交影响力是用户交互行为的内在诱因，而交互行为是社交影响力的外在表现，从而对信息的传播产生直接影响。社交影响力是社会影响力在线社交网络中的自然延伸，而社会影响力被认为是个人行为能够直接或间接地影

收稿日期：2017−11−21；**修回日期**：2018−08−22
通信作者：全拥，qy8801@nudt.edu.cn
基金项目：国家重点研发计划基金资助项目（No.2017YFB0803303）；国家自然科学基金资助项目（No.61502517）；湖南省重点研发计划资助项目（No.2018GK2056）
Foundation Items: The National Key Research and Development Program of China (No.2017YFB0803303),The National Natural Science Foundation of China (No.61502517), The Key Research and Development Project of Hunan Province (No.2018GK2056)

响他人的想法、感情以及行动[2]。因此，社交影响力可以通过用户之间的社交活动体现出来，表现为在线社交网络中用户的行为和思想等受他人影响发生改变的现象[3]。

影响力分析是在线社交网络分析的重要内容，在舆情引导与社会运作中起着重要作用，具有广泛应用，例如信息推荐[4]、专家发现[5]、影响极大化[6]、病毒式营销[7]等。作为社交影响力分析的主要内容，个体影响力度量一直是学术界的研究热点问题，主要是定量计算个体的影响力大小，通过排名技术发现在线社交网络中的影响力个体。影响力个体在不同应用中又可被称为意见领袖[8]、领域专家[5]等。最初，相关学者在社会网络中发现了人们的影响力存在差异性，即具有广泛影响力的个体更容易将自己的观点传达给其他人。同样，在线社交网络中的影响力用户发布或评论的信息，更容易引发大量用户的转发和阅读，如新浪微博中的大 V 用户。因此，在线社交网络中的影响力个体在创新采用、网络群体聚集、信息传播与导向等方面发挥着重要作用。但是，由于理论模型和实验方法的限制，早期工作只能从小样本数据集上定性地分析个体影响力，验证了社会系统中个体影响力的存在性。在线社交网络提供了丰富可用的实验数据，研究者可以对用户本身体现出来的影响力进行建模和量化计算[9-11]。

实际上，由在线社交网络数据构建的图结构模型相当复杂，一般包含上亿个用户节点、用户之间关系构成的成百上千亿条边以及他们产生的海量网络信息，如截至 2017 年 6 月底，新浪微博的活跃用户数已达 3.65 亿。这对个体影响力计算方法提出了新的挑战，难以在如此超大规模图上高效度量在线社交网络用户的影响力。但是，不同类别大数据处理框架的出现使高效分析上述海量数据成为可能。首先，基于采用的小样本数据集或子图结构，对个体影响力度量模型进行分析和验证。然后，结合具体的大数据处理框架，对个体影响力度量模型进行并行化实现。最后，在集群环境中部署个体影响力并行化算法，高效地计算在线社交网络用户的影响力[12-14]。当前，Apache 基金会开发的一种开源的分布式基础框架 Hadoop 应用比较广泛，它实现了一个用于存储海量数据的分布式文件系统（HDFS）。基于 Hadoop 平台，本文选取 MapReduce 和 Spark 两种并行计算模型来说明大数据处理框架对个体影响力度量算法的性能影响。针对真实的大规模在线社交网络数据和不同的大数据处理框架下，实验利用上述并行编程模型分别实现了一类经典的个体影响力度量算法，并对不同规模数据集以及不同集群之间的算法性能进行了比较。

2 个体影响力计算方法

在线社交网络个体影响力度量算法主要从网络结构、用户行为、交互信息等方面对用户自身表现出的社交影响力进行建模分析及量化计算。一般地，在线社交网络的拓扑结构可由图 $G=\{V,E\}$ 表示，其中，V 是用户集合，E 是用户之间的关系构成边的集合。实际应用中，当用户之间的关系是有向的，那么 G 是有向图，如转发关系；当用户之间的关系是无向的，那么 G 是无向图，如好友关系。G 也可以是带权图，ω_{uv} 表示用户 u 和用户 v 之间形成边的权重，如转发频率和好友亲密度等。早期的个体影响力计算方法主要在拓扑结构图中利用复杂网络的相关概念来定量计算在线社交网络中用户的影响力，如图中节点的出度与入度、度中心度[15]、接近中心度[16]、介数中心度[17]、K–壳[18]等。这些方法表达的意义比较直观，被广泛应用于在线社交网络中用户影响力的分析。例如，节点的出度与入度直接衡量了用户对其邻居用户的影响力；度中心度衡量了用户对其邻居用户的平均影响力；接近中心度衡量用户对其他用户的间接影响力；介数中心度和 K–壳都衡量了用户在信息传播扩散过程中的影响力。但是这类基于网络结构的方法也有其局限性，没有充分考虑用户自身行为或用户之间的交互信息等数据，导致最终计算的用户影响力结果不够精确。

为了准确度量在线社交网络中用户的社交影响力，相关学者借鉴了经典的网页排名模型 PageRank 算法[19]，通过融合用户属性和网络信息等因素，设计了多种个体影响力度量算法。PageRank 算法最初被应用于 Google 的搜索引擎中，是一种基于反向链接和正向链接分析的网页排名算法。该算法利用一种基于马尔可夫的随机游走思想来模拟用户浏览网页的行为，并认为一个网页的重要性由所有链向它网页的重要性决定。假设 $G=\{V,E\}$ 是由互联网中所有的网页及其链接关系构成的图结构，$\boldsymbol{P}$ 为网页得分组成的向量，$\boldsymbol{M}$ 是由链接关系产生的转移概率矩阵，则 PageRank 算法可用矩阵乘

积的形式为

$$\boldsymbol{P} = \alpha \boldsymbol{M}^{\mathrm{T}} \boldsymbol{P} + \Delta \tag{1}$$

其中，α为正则化因子，Δ是修正项。最初的PageRank算法没有修正项，$\boldsymbol{P}$是$\alpha\boldsymbol{M}^{\mathrm{T}}$的特征向量，网页排名的过程等同于求解主特征向量的过程。不难看出，式(1)是一个迭代算法，其时间复杂度为$O(|E|^2)$。在实际应用中，为了保证算法的收敛性，可令$\Delta = \beta\boldsymbol{e}$，$\boldsymbol{e}$是元素全为1的向量，$\beta$是调节因子。

基于 PageRank 算法，在线社交网络中个体影响力分析可以转化为在用户之间关系构成的图中挖掘用户的影响力。此时，用户影响力的传播过程也是一种随机游走过程，可以利用随机游走路径上的用户衡量当前用户的影响力，即用户影响力与和他相连的其他用户的影响力相关，其他用户的影响力越大，则当前用户的影响力越大，反之亦然。文献[20]利用 Twitter 中用户之间的关注关系构造了转移概率矩阵 $\boldsymbol{M}$，并令式(1)中$\Delta = \dfrac{(l)\boldsymbol{e}}{N}$（$N$为用户规模）来计算用户影响力。该方法的思想是用户的影响力由他粉丝的影响力决定，粉丝影响力越大且越少关注其他用户，则粉丝对该用户的影响力的贡献越大。

为了更细粒度的度量在线社交网络中用户的影响力，相关学者以式(1)为基础，提出了结合用户特征和交互信息的个体影响力计算方法。具体地，就是根据已有的在线社交网络数据重新构造式(1)中的转移概率矩阵 $\boldsymbol{M}$ 和修正项Δ。例如，文献[21]提出了考虑用户特征的个性化 PageRank 算法，即$\Delta=(1-\alpha)\boldsymbol{r}$，$\boldsymbol{r}$ 表示用户对话题的偏好程度以及发布信息的新颖度等。文献[12]通过构造话题相关的影响力矩阵 $\boldsymbol{M}$，在不同话题层面上计算用户的影响力。文献[11]在 PageRank 算法基础上考虑话题相似性，提出了 TwitterRank 算法。文献[22]提出了结合用户发布信息特征的 InfluenceRank 算法。文献[10]提出了基于多关系网络转移矩阵 $\boldsymbol{M}$ 的微博用户影响力度量模型。由此可见，PageRank 算法是在线社交网络个体影响力度量的基础算法。本文选取式(1)作为测试算法，通过用户之间的转发关系构造转移概率矩阵 $\boldsymbol{M}$ 以及令$\Delta = \dfrac{(1-\alpha)\boldsymbol{e}}{N}$，并在大规模真实数据集中计算用户的影响力。最后，在不同规模集群环境下，对比两种大数据处理框架下的算法性能。

3　算法并行化实现

大数据处理框架为处理和分析在线社交网络大规模数据提供了技术支持，研究人员可以将已有的在线社交网络个体影响力算法与具体的大数据并行计算框架相结合，用于分析用户的影响力。可以看出，上述 PageRank 算法及其改进算法的时间复杂度依然是$O(|E|^2)$。针对在线社交网络用户及其关系构成的海量数据时，传统的单机串行算法使内存、CPU、I/O 等硬件资源无法满足需要。通过 MapReduce 和 Spark 两种并行计算框架对改进的 PageRank 算法实现并行化编程，提高算法的执行效率。

3.1　基于 MapReduce 并行框架

MapReduce 是由 Google 公司提出的一种面向大规模数据处理的并行计算框架。它被分为 map 处理阶段和 reduce 处理阶段，并且每个阶段的输入和输出都可以自定义数据类型的键值对格式。实际应用中，开发人员需要指定 map 函数和 reduce 函数来实现相应算法的不同功能，而不需要关注分布式底层实现机制。MapReduce 程序执行时，每个 map 操作都是并行运行且相互独立的，但可能会受到数据源和 CPU 等硬件资源的影响。同样地，多个 reduce 操作执行时，所有具有相同键值的 map 输出会聚集到同一个 reduce 中。在执行 map 操作之前，大数据将会被分割成若干小数据块，通过 map 函数处理完后会产生一系列键值对。这些键值对按键值进行排序和合并，接着把整理好的数据输入到多个 reduce 中，每个 reduce 操作对已经排好序的并且带有相同键值的输入数据进行迭代计算，最后把结果输出到 HDFS 中。MapReduce 并行框架的另一个特点是并行处理时可以提供部分容错和出错恢复的功能，如当一个 map 操作或 reduce 操作失效时，作业会被重新安排，从而保证作业连续执行。

本文基于 MapReduce 计算框架对式(1)实现了并行化编程，主要是重写 map 函数和 reduce 函数，伪代码如算法 1 所示。显然，该并行算法是迭代算法，当不满足迭代终止条件时，算法每一次的迭代操作都相同：map 操作负责将每个用户的影响力按权重比传播给其他相关用户，而 reduce 操作负责搜集各影响力分量并根据式(1)更新当前用户的影响力值。

算法 1　基于MapReduce的个体影响力度量算法

输入 带权重的在线社交网络结构图 $G = \{V, E, \boldsymbol{W}\}$，正则化因子$\alpha$

输出 在线社交网络用户的社交影响力值 $\boldsymbol{P}$

1) 计算转移概率矩阵 $\boldsymbol{M} = \{m_{uv}\}$；

2) 初始化 $N = |V|$，$\boldsymbol{P} = \frac{1}{N}$，$\alpha$；

3) repeat：

4) map：

5) for each $v \in V$ do

6) for each$(v, u) \in E$ do

7) 计算影响力传播分量 $P_{u\to v}=m_{uv}\times P(u)$；

8) end

9) end

10) reduce：

11) for each $v \in V$ do

12) $P'(v) = 0$；

13) for each $(v, u) \in E$ do

14) 影响力分量线性加权 $P'(v)=P'(v)+\alpha P_{u\to v}$；

15) end

16) Update $P(v) = P'(v) + \frac{1-\alpha}{N}$；

17) end

18) until convergence；

19) for each $v \in V$ do

20) 输出已收敛的用户影响力值 $P'(v)$；

21) end

3.2 基于 Spark 并行框架

Spark 是由加州大学伯克利分校 AMP 实验室开发的通用内存并行计算框架。它的主要思想是通过一种新的作业和数据容错方式来减少磁盘和网络的 I/O，从而提高海量数据的处理效率。弹性分布式数据集 RDD 是 Spark 的核心技术，表示已被分片、不可变地被并行操作的数据集合。RDD 是对计算和数据的抽象，拥有方便重建的容错机制并提供了转换和动作两大类算子。转换算子负责将一个或多个 RDD 转换成新的 RDD，动作算子则根据生成的 RDD 产生最终的计算结果。Spark 应用提交后，外部数据经过一系列转换算子形成 RDD；动作算子触发作业提交，根据 RDD 之间的依赖关系创建有关所有操作的有向无环图 DAG 计算模型；DAGScheduler 解析 DAG 图并将构建不同的 Stage，由任务调度器将 Stage 分解的任务集提交到集群节点中运行。

基于内存计算的 Spark 并行框架适用于迭代算法，它的运行模式有多种，不同运行模式具有相似的运行流程，只是资源分配模式和任务调度模块有所不同。本文结合 Spark 并行计算框架并行化实现了式(1)，并在 Yarn 运行模式进行测试，伪代码如算法 2 所示。类似于算法 1，算法 2 也是迭代算法，每一次迭代的操作都相同：将在线社交网络图结构等数据转化成 RDD 格式数据集，flatmap()算子负责扩散用户的影响力，reducebykey(*add*)算子累加各影响力分量，map()算子依照式(1)更新当前用户的影响力值。

算法 2 基于 Spark 的个体影响力度量算法

输入 带权重的在线社交网络结构图 G $\{V, E, \boldsymbol{W}\}$，正则化因子α

输出 在线社交网络用户的社交影响力值 $\boldsymbol{P}$

1) 计算转移概率矩阵 $\boldsymbol{M} = \{m_{uv}\}$；

2) 初始化 $N = |V|$，$\boldsymbol{P} = \frac{1}{N}$, α；

3) RDD (V, E, $\boldsymbol{M}$): = SparkContext (G, $\boldsymbol{M}$). SparkOperator；

4) repeat：

5) for each $v \in V$ do

6) for each $(v, u) \in E$ do

7) 计算影响力传播分量 RDD(v, $P_{u\to v}$): RDD(V, E, $\boldsymbol{M}$). flatmap(lamda: $P_{u\to v} = m_{uv}\times P(u)$)；

8) 用户影响力值更新 RDD(v, P_v) : RDD(v, $P_{u\to v}$).reducebykey(*add*).mapvalue(lamada:$P(v)=\alpha P(v)+\frac{1-\alpha}{N}$；

9) end

10) end

11) until convergence；

12) for each $v \in V$ do

13) 输出已收敛的用户影响力值 $P'(v)$；

14) end

4 实验结果与分析

为了测试大数据处理框架对在线社交网络个体影响力度量算法性能的影响，本文通过编程实现了上述两种并行算法，并在真实大规模数据集上对比分析了算法的相关性能。

4.1 实验数据及预处理

实验数据集是通过湖南蚁坊软件股份有限公

司的爬虫系统获取的，主要利用新浪微博 API 接口搜集了新浪微博平台注册用户在 2016 年 11 月 2 日至 2017 年 6 月 26 日期间产生的真实博文数据。每条博文数据是一个文本记录，包括 5 个字段：时间戳、用户 ID、博文 ID、转发用户 ID、转发博文 ID。当用户发布某条原创博文时，该博文数据的转发用户 ID 和转发博文 ID 字段就都为 null。该数据集共涉及在线社交网络平台 116 147 966 位用户产生的 4 586 584 659 条博文，其中，原创博文 1 079 801 756 条，转发博文 3 506 782 903 条，具体统计信息如表 1 所示。

表 1　　实验数据集统计信息

数据集描述	统计量
博文数	4 586 584 659
用户数	116 147 966
原创博文数	1 079 801 756
被转发原创博文数	97 351 945
转发博文数	3 506 782 903
原创博文平均被转发次数	3.25

可以看出，不足 10%的原创博文被其他用户转发，且只有约 0.2%的原创博文被转发超过 500 次，这说明在线社交网络中少量用户产生并控制着大量信息的传播。图 1 展示了上述数据集中原创博文被转发次数的概率分布。该分布具有明显的无标度特性，符合指数为 2.13 的幂律分布，即少量原创博文存在较多的转发次数。

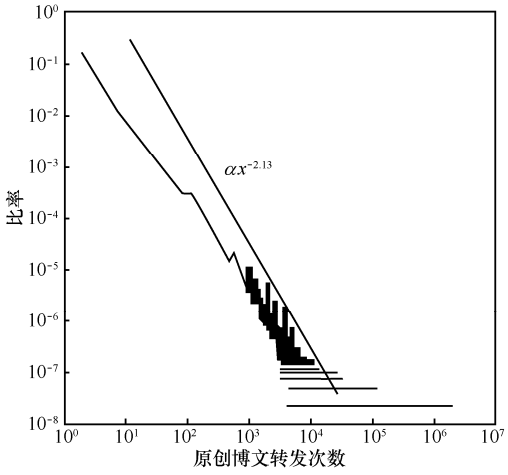


图 1　原创博文转发次数概率分布

个体影响力度量需要以特定的网络结构为基础进行计算，本文通过提取上述数据集中用户之间的转发关系来构建带权重的在线社交网络结构图。具体操作如下：首先，针对任何一条博文数据，抽取用户 ID 和转发用户 ID 两个字段数据；然后，删除其中转发用户 ID 为 null 的转发关系数据；最后，合并具有相同转发关系的数据，得到具有转发频次的三元组转发关系数据集，即<用户 ID, 转发用户 ID, 频次>。为了保护新浪微博用户的隐私，需要对用户 ID 进行去隐私化处理，最终得到约 59 GB 的转发关系数据集 R。若在数据集 R 中存在三元组$<u, v, f_{u\to v}>$，则表示用户 u 在样本数据集中转发用户 v 的相关博文共计 $f_{u\to v}$ 次。数据集 R 共包含 3 504 379 868 个转发关系三元组，涉及 115 205 577 位用户，存储在块大小为 128 MB 的 HDFS 文件系统中。

基于转发关系数据集 R，可以构建在线社交网络转发关系结构图 $G = \{V, E, \boldsymbol{W}\}$。其中，$V$ 表示用户集合，E 表示用户之间转发关系构成的边集合，$\boldsymbol{W}$ 代表相应的边权重矩阵。在本实验中，$|V| = 115\,205\,577$，$|E| = 3\,504\,379\,868$。针对数据集 R 中的任意三元组$<u, v, f_{u\to v}>$，存在$(v, u)\in E$，且 $w_{v,u} = f_{u\to v}$。

图 2 左边是由用户转发关系三元组数据子集构成的网络结构图示例，节点代表不同的用户，边的方向代表了博文转发路径，边的权值代表用户之间的转发频次。如用户 u_2 共 w_{u_1,u_2}=4 次转发过用户 u_1 的博文。在线社交网络用户的个体影响力以博文信息为载体，沿着转发关系网络进行扩散。因此，个体影响力的扩散路径和概率可以由带权重的用户转发关系网络图计算得到，即对任意的边$(v, u) \in E$，存在从用户 u 到用户 v 影响力扩散路径，且扩散概率为

$$m_{uv}=\frac{w_{v,u}}{\sum_{v'\in V}}, (v',u)\in E\ w_{v',u} \tag{2}$$

图 2 右边所示是基于在线社交网络用户转发关系网络图计算影响力扩散概率的示例，虚线表示影响力扩散路径，边上的概率值是对应的影响力扩散概率。实际应用中，通过式(2)可以计算出算法 1 和算法 2 中所需的转移概率矩阵 $\boldsymbol{M}$。

4.2　实验环境

本实验中，算法 1 和算法 2 两种并行算法及对数据的预处理都是通过 Java 语言编程实现，使用的开发工具包是 JDK1.8。实现的并行算法程序运行在由腾讯云服务器搭建的Hadoop分布式集群环境中，

Hadoop 版本为 2.7.4，Spark 版本为 1.6.2。该集群共由 128 个独立的内存型 M2 服务器节点组成，每个节点的硬件配置如下：8 核 CPU，64 GB 内存，500 GB 硬盘，1 Mbit/s 带宽，预装系统版本为 Ubuntu Server 14.04.1 LTS 64 位。为了对比算法在不在规模集群上的性能，本实验分别搭建了 6 种不同规模的集群环境，其唯一区别是具有不同的服务器节点数目，分别是 4、8、16、32、64、128。在不同集群环境中，其中，只有一台服务器作为主节点，其他服务器均是数据节点或计算节点。本文实现并在单机环境下运行了在线社交网络个体影响力测试算法，其使用的机器也是腾讯云提供的内存型 M2 服务器。

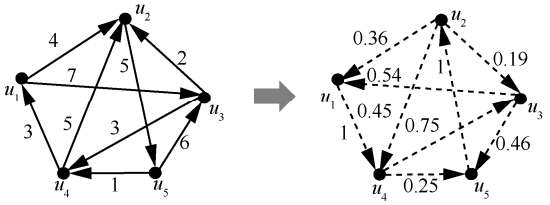


图 2　影响力扩散示例

4.3　性能指标与测试参数

针对在线社交网络个体影响力度量算法，本实验主要从准确度和运行时间等指标对基于式(1)的并行化算法 1 和算法 2 在真实数据上进行相关性能测试。

准确度方面。由于缺少针对大规模用户影响力值计算的标准测试数据集，本实验主要从算法收敛情况对准确度进行测试。式(1)本质上是幂迭代算法，因此算法 1 和算法 2 在实际运行过程中需要预先设置终止条件。当程序运行达到终止条件时，算法迭代结束，实验将会记录此时的迭代次数和迭代条件变化情况，具体地给定第 n 次迭代后计算的用户影响力值向量 $\boldsymbol{P}^n$，当式(3)成立时，程序终止，算法收敛。

$$\frac{\| \boldsymbol{P}^{n+1} - \boldsymbol{P}^n \|_1}{N} < \delta \tag{3}$$

其中，N 表示实验数据集中的用户数，误差限 $\delta=10^{-8}$。$\|\cdot\|_1$ 表示向量的 1−范数，即向量所有元素的绝对值之和。当 $n = 0$ 时，$\boldsymbol{P}^0$ 表示初始化的用户影响力值。式(3)认为平均每个用户的影响力数值误差不超过 10^{-8} 时，计算结果趋于稳定，算法已收敛。

运行时间方面。通过设置不同参数，实验将记录算法在不同数据集以及不同分布式集群上的运行时间和加速比。基于式(1)的在线社交网络个体影响力度量算法只需设置一个参数，它是正则化因子 α，也称为跳转因子。在进行大规模网页排名计算时，α通常取值为 0.85，表示上网者按照链接浏览网页的概率为 0.85，随机跳转到一个新网页的概率为 0.15。在线社交网络用户的转发行为不同于上网者随机点击页面的过程，因此本实验测试了并行算法在α值分别为 0.5、0.7、0.85 和 0.95 时的性能。

并行化算法在不同数据集上的性能存在差异，为了探究大数据处理框架在不同数据集上的加速效果，本实验基于数据集 R 划分了不同规模的数据子集 D_1、D_2、D_3、D_4、D_5，具体描述如表 2 所示。这些数据集涉及的在线社交网络用户规模从十万级至亿级递增，用户之间形成的转发关系数也相应增加，最多达到十亿级规模。

表 2　　实验数据集子集描述

数据集	用户数	转发关系数	稠密度
D_1	100 000	4 185 326	4.19×10^{-4}
D_1_A	100 000	41 862 541	4.19×10^{-3}
D_1_B	100 000	418 568	4.19×10^{-5}
D_2	1 000 000	94 091 124	9.41×10^{-5}
D_2_A	1 000 000	940 634 687	9.41×10^{-4}
D_2_B	1 000 000	9 411 885	9.41×10^{-6}
D_2_C	1 000 000	941 482	9.41×10^{-7}
D_3	10 000 000	662 538 337	6.63×10^{-6}
D_4	50 000 000	1 368 256 085	5.47×10^{-7}
D_5	115 205 577	3 504 379 868	1.19×10^{-4}

在线社交网络密度用于刻画中节点间连边的密集程度，定义为图 $G = \{V, E, \boldsymbol{W}\}$的邻接矩阵中非零元素所占比例，在此又称稠密度。具有相同用户数的在线社交网络数据集，拓扑图的稠密度也会由于用户之间转发关系数的不同而有所差异。本实验以数据子集 D_1 为样本，通过随机采样增加或减少节点间边的方法构造了具有不同稠密度的数据子集 D_1_A、D_1_B、D_2_A、D_2_B、D_2_C，如表 2 所示。通过对上述数据子集进行实验，可以探究稠密度对算法性能的影响。

4.4　结果与分析

本文依照上节中指标要求和参数设置，在不同真实数据子集上对在线社交网络个体影响力并行

化算法 1 和算法 2 进行了相关性能测试。由于目前不存在针对在线社交网络用户影响力计算的标准测试数据，所以从收敛性和计算效率两方面对本实验结果进行分析，具体实验结果及其对比情况如下所述。

4.4.1 收敛性分析

由于算法 1 和算法 2 都是基于式(1)的并行化实现，因此在不同的大数据处理框架下算法具有相同的收敛情况。本节将以基于 Spark 的并行化算法 2 为例，在不同参数配置环境中阐述算法在不同数据子集上的收敛性能，基于 MapReduce 的并行化算法 1 的情况类似。

实验结果表明，给定 α 值，同一数据子集在不同规模集群上的收敛情况相同，数据子集 D_1、D_2、D_3、D_4、D_5 第一次满足收敛条件（$\delta=10^{-8}$）时，算法的迭代次数分别是 83、84、84、84、85。这是因为集群规模的变化只会引发计算资源的变化，不会改变算法的运行原理。图 3 是在 16 节点集群下，α=0.85 时算法 2 在不同数据子集中的收敛趋势变化。在迭代初期，收敛速度较快，随着程序运行，收敛速度逐渐变慢。当收敛条件相同时，基于式(1)的在线社交网络个体影响力度量算法的收敛速度与用户规模无关。

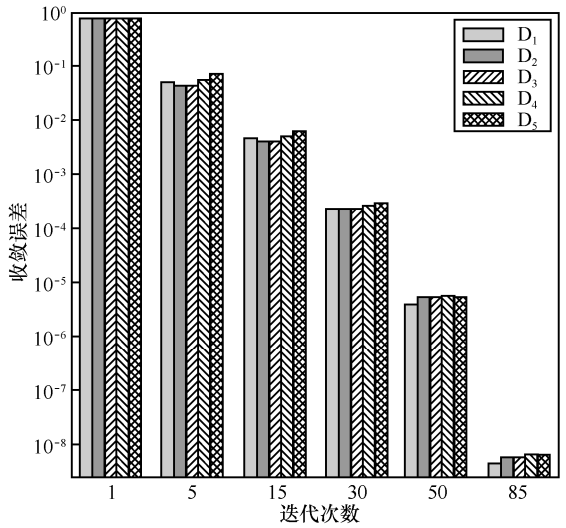


图 3　算法 1 在不同数据子集中的收敛变化情况

在 64 节点集群上，基于不同α值的算法 2 在数据子集 D_4 上的收敛变化情况如图 4 所示。可以看出，随着α值的增加，算法 2 的收敛速度变慢。当α取值依次为 0.5、0.7、0.85 时，算法 2 收敛所需迭代次数分别是 25、44、84。当α=0.95 时，算法 2 迭代第 212 次时的收敛误差为 2.2×10^{-8}，此时仍未满足收敛条件。由此可见，在线社交网络用户倾向于转发好友的博文时，α取值偏高，个体影响力度量算法的收敛速度越慢。在实际应用中，算法应根据具体的在线社交网络平台用户行为特征，设置合理的跳转因子α参数进行计算。

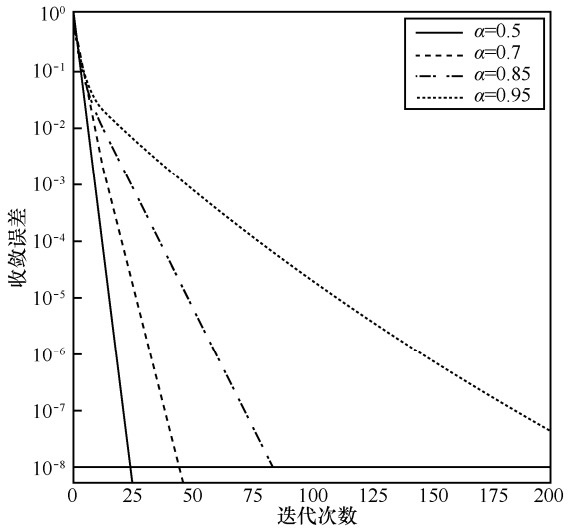


图 4　算法 2 在不同α值时的收敛变化情况

基于不同稠密度的数据子集 D_1、D_1_A、D_1_B，在 8 节点集群上，图 5 展示了算法 2 在α=0.85 时收敛情况。当满足收敛条件（$\delta=10^{-8}$）时，算法的迭代次数分别为 83、79、70。这说明在线社交网络个体影响力度量算法的收敛速度与用户之间关系构成的图稠密度有关，通过构建具有不同稠密度的概率转移矩阵，基于式(1)的个体影响力计算方法的收敛性能会有所差异。

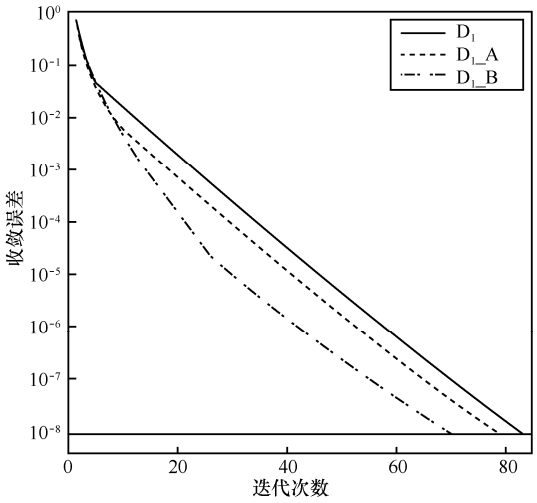


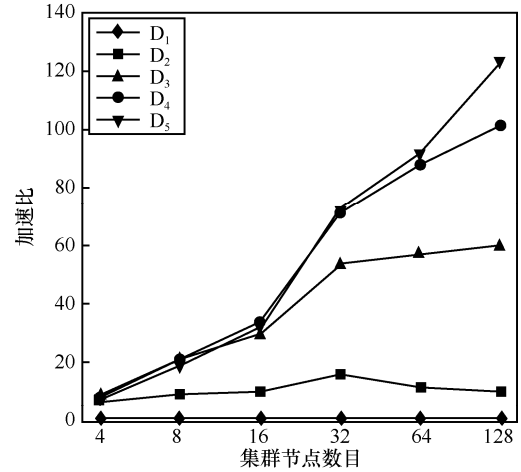
图 5　算法 2 在不同稠密度数据子集中的收敛变化情况

4.4.2　效率分析

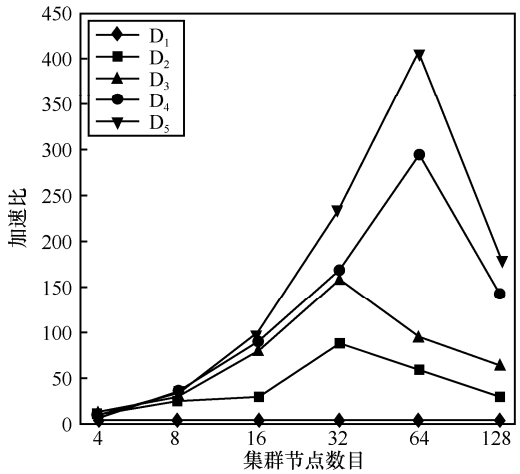
大数据处理框架的特点是可以提高算法处理数据的能力，基于大数据处理框架的并行化算法在不同参数配置环境下具有不同的计算效率。由于在拥有少量服务器节点的集群环境中，处理大规模数据集所需时间较长，因此在进行效率分析时，算法的终止条件是迭代达到预设次数，而不是收敛误差小于预设阈值。接下来，本文将从多个角度对比分析基于 Spark 和 MapReduce 框架并行化程序的运行效率。

一般采用加速比衡量并行化程序的性能和效果，它是指同一个任务在单处理器系统和并行处理器系统中运行消耗时间的比率。图 6 显示了当 α=0.85 时，算法 1 和算法 2 在具有不同服务器节点数目集群中的加速比情况，其中，程序在数据子集 D_1、D_2、D_3、D_4、D_5 中的运行迭代次数分别设置为 50、40、30、20、20。总体而言，在相同情况下，基于 Spark 框架的并行化算法的加速比要高于基于 MapReduce 框架的并行化算法。这是因为 Spark 是基于内存进行的迭代计算，其带来的性能提升更大。当然，集群中服务器节点数目的增多对于两种并行化算法都具有一定的加速作用，且在大规模数据子集上效果更明显。这是因为随着服务器节点数目增多，可用的计算资源越多，算法运行效率越高；随着数据子集规模增大，计算资源利用更充分，带来的加速效果更明显。

但在有些情形下，当集群节点数目继续增多时，并行化算法的加速比反而减小。如图 6(b)所示，算法 2 在每个数据子集中都有一个最高加速比，其对应的集群节点数目分别是 8、32、32、64、64。这些集群节点数目又称为算法在该数据子集下加速比曲线的性能拐点。当集群中节点数目超过其拐点时，由于并行化模型的限制和大数据处理框架的特征，算法的加速性能不会随着集群节点的增多而继续提高。图 6(a)展示了算法 1 在数据子集 D_1、D_2 中的性能拐点分别是 8、32，而在其他数据子集中并未出现性能拐点。这说明通过增加集群中服务器节点数量，算法 1 在数据子集 D_3、D_4、D_5 中获得的加速比会持续增大。此外，算法 1 在数据子集 D_1 中的加速比均小于 1。由于该数据子集规模小，分布式环境中服务器节点间的通信、子任务的创建、数据块分发等消耗的时间大于算法 1 相较于串行算法节省的运行时间。



(a) 算法1在不同数据子集中加速比



(b) 算法2在不同数据子集中的加速比

图 6　不同规模集群环境中算法的加速比

在 128 服务器节点集群环境中，当收敛条件都满足式(3)且 $\delta=10^{-8}$、α=0.85 时，图 7 显示了算法 1 和算法 2 在不同规模数据子集中的运行时间。

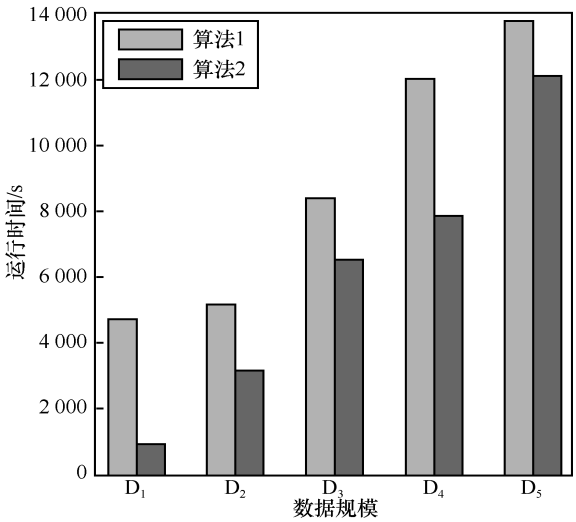


图 7　算法在不同规模数据子集中的运行时间

显然，随着数据集规模的增长，算法 1 和算法 2 的运行时间都呈现递增趋势，且在不同数据子集中，算法 2 的执行时间显著少于算法 1 的执行时间。此外，结合上述分析，当性能拐点出现后，两种并行化算法的运行时间之差逐渐缩小。

以数据子集 D_3 为例，图 8 所示是算法 1 和算法 2 在不同规模集群环境中迭代运行 30 次的单次平均迭代时间及其方差，此时$\alpha = 0.85$。可以看出，随着集群节点数目的增多，算法单次迭代所需时间更少，这与图 6 中结果一致。当算法 2 出现性能拐点（集群节点数目 32）后，其单次迭代时间开始增长。算法 1 单次迭代时间的方差要大于算法 2，这说明该实验中 Spark 并行框架的计算效率更稳定。

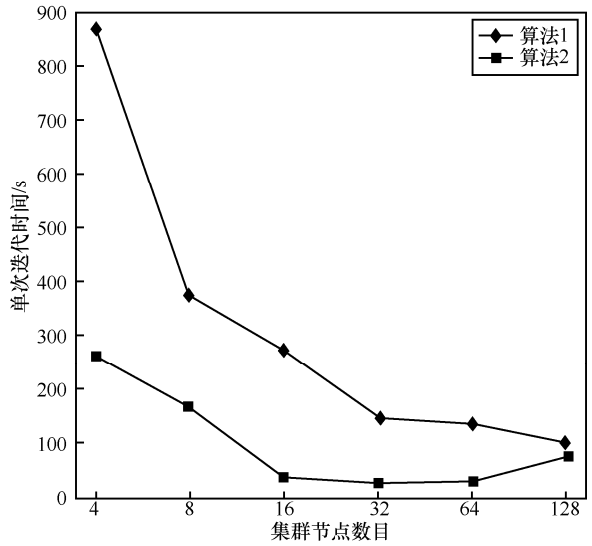


图 8　不同规模集群环境中算法迭代一次的时间

在具有 64 节点服务器的集群环境中，图 9 展示了当α取不同值时，两种并行化算法在数据子集 D_4 上迭代计算 20 次的单次平均迭代时间及其方差。可以看出，算法 1 完成一次迭代计算所需时间更长。当α=0.95 时，算法 1 和算法 2 的单次迭代所需时间最长，而当α=0.85 时，算法 1 和算法 2 的单次迭代所需时间最短。这说明α值的选取对基于式(1)的在线社交网络个体影响力度量算法的计算效率具有直接影响。

图 10 是在 16 节点集群环境中，算法 1 和算法 2 针对具有相同规模不同稠密度的数据子集 D_2、D_2_A、D_2_B、D_2_C，迭代运行 40 次的单次平均迭代时间及其方差，此时，α=0.85。不难看出，随着相同规模数据子集稠密度的增加，算法 1 和算法 2 完成单次迭代所需时间更长，且它们的方差也在变大。这说明在计算在线社交网络用户的个体影响力时，不仅用户数规模会直接影响算法的效率，而且用户间关系构建的网络图密度也会影响算法的计算效率。

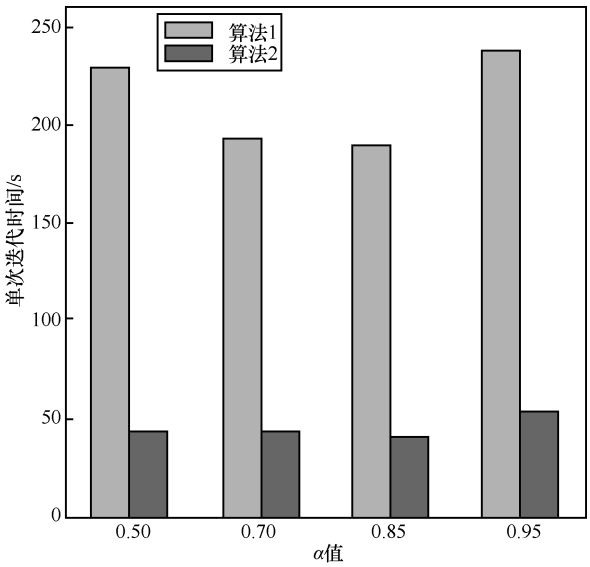


图 9　不同α值时算法迭代一次的时间

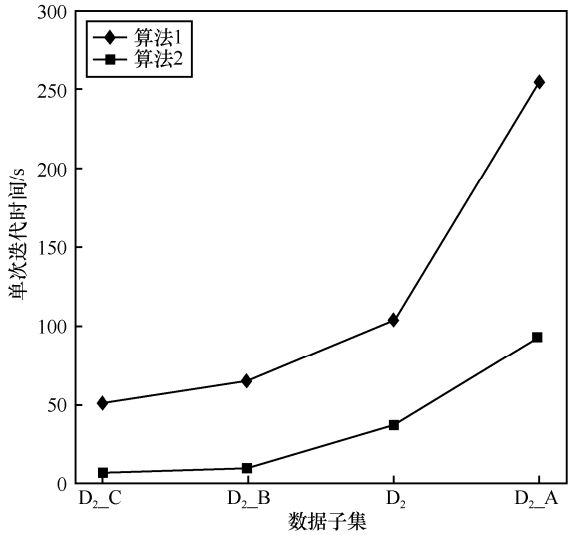


图 10　算法在不同稠密度数据子集中迭代一次的时间

5　结束语

本文主要基于一种经典的在线社交网络个体影响力算法，结合 MapReduce 和 Spark 两种并行计算框架，在真实大规模新浪微博数据集上进行了性能测试。实验结果表明，大数据处理框架能够对在线社交网络个体影响力算法的效率产生显著影响。MapReduce 和 Spark 由于内在并行机制的差异，导致算法处理大数据集时的性能也会存在差别。在实际使用过程中，多种参数的设置和实验数据集的特

征对算法的收敛性和计算效率有直接影响。由于数据集规模的不同，大数据处理框架对算法在集群计算过程中带来的加速性能不同。在线社交网络用户之间关系构建的图结构越稠密，其计算复杂度越高，算法迭代次数和运行时间会更多。

本文只对个体影响力度量算法进行了简单的并行化实现，在实验过程中大数据处理框架相关参数采用默认配置，主要是为了测试文中算法在大规模社交网络数据中的性能，以及对后续个体影响力算法的设计和并行化实现提供实证参考依据。因此，进一步工作可以通过优化大数据处理框架的相关参数提高在线社交网络个体影响力并行化算法的性能。

参考文献：

[1] 方滨兴, 许进, 李建华. 在线社交网络分析[M]. 北京: 电子工业出版社, 2014.
FANG B X, XU J, LI J H. Online social network analysis[M]. Beijing: Publishing House of Electronics Industry, 2014.

[2] CIALDINI R B. Influence: science and practice[M]. Boston: Allyn and Bacon, 2003.

[3] 吴信东, 李毅, 李磊. 在线社交网络影响力分析[J]. 计算机学报, 2014, 37(4):735-752.
WU X D, LI Y, LI L. Influence analysis of online social networks[J]. Chinese Journal of Computers, 2014, 37(4):735-752.

[4] TING I H, CHANGP S, WANG S L. Understanding microblog users for social recommendation based on social networks analysis[J]. Journal of Universal Computer Science, 2012, 18(4):554-576.

[5] LI N, GILLET D. Identifying influential scholars in academic social media platforms[C]//The 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. 2013: 608-614.

[6] VEGA-OLIVEROS D A, BERTON L, LOPES A D A, et al. Influence maximization based on the least in-fluential spreaders[C]//The 1st International Conference on Social Influence Analysis. 2015: 3-8.

[7] DINH T N, ZHANG H, NGUYEN D T, et al. Cost-effective viral marketing for time-critical campaigns in large-scale social networks[J]. IEEE/ACM Transactions on Networking, 2014, 22(6):2001-2011.

[8] KATZ E, LAZARSFELD P. Personal influence: the part played by people in the flow of mass communica-tions[M]. New Jersey: Transaction Publishers, 1966.

[9] CHA M, HADDADI H, BENEVENUTO F, et al. Measuring user influence in twitter: the million follower fallacy[C]//International Conference on Weblogs and Social Media.2010: 10-17.

[10] DING Z, JIA Y, ZHOU B, et al. Mining topical influencers based on the multi-relational network in microblogging sites[J]. China Communications, 2013, 10(1):93-104.

[11] WENG J, LIM E P, JIANG J, et al. TwitterRank: finding topic-sensitive influential twitterers[C]//The third ACM International Conference on Web Search and Data Mining. 2010: 261-270.

[12] TANG J, SUN J, WANG C,et al. Social influence anal-ysis in large-scale networks[C]//The 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2009: 807-816.

[13] LIU X, LI M, LI S, et al. IMGPU: GPU-accelerated influence maximization in large-scale social networks[J]. IEEE Transactions on Parallel and distributed Systems, 2014, 25(1):136-145.

[14] 平宇, 向阳, 张波, 等. 基于 MapReduce 的并行 PageRank 算法实现[J]. 计算机工程, 2014, 40(2):31-34.
PING Y, XIANG Y, ZHANG B, et al. Implementation of parallel PageRank algorithm[J]. Computer Engineering Based on MapReduce, 2014, 40(2):31-34.

[15] FREEMAN L C. Centrality in social networks conceptual clarification[J]. Social Networks, 1978, 1(3):215-239.

[16] NEWMAN M E J. A measure of betweenness centrality based on random walks[J]. Social Networks, 2005, 27(1):39-54.

[17] NEWMAN M E J. The structure and function of complex networks[J]. SIAM Review, 2003, 45(2):167-256.

[18] KITSAK M, GALLOS L K, HAVLIN S, et al. Identification of influential spreaders in complex networks[J]. Nature Physics, 2010, 6(11): 888-893.

[19] PAGE L, BRIN S, MOTWANI R, et al. The pagerank citation ranking: bringing order to the web[J]. Stanford Digital Libraries Working Paper, 1998, 9(1):1-14.

[20] EFRON M. Information search and retrieval in microblogs[J]. Journal of the American Society for Information Science and Technology, 2011, 62(6): 996-1008.

[21] HAVELIWALA T, KAMVAR A, JEH G. An analytical comparison of approaches to personalizing pagerank[R]. Palo Alto: Stanford University, 2003.

[22] SONG X, CHI Y, HINO K, et al. Identifying opinion leaders in the blogosphere[C]//The 6th ACM Conference on Information and Knowledge Management. 2007: 971-974.

[作者简介]

全拥（1988−），男，湖南常德人，国防科技大学博士生，主要研究方向为在线社交网络分析、数据挖掘。

贾焰（1960−），女，四川成都人，博士，国防科技大学教授、博士生导师，主要研究方向为数据挖掘、大数据分析、信息安全等。

张良（1989−），男，江西九江人，国防科技大学博士生，主要研究方向为在线社交网络分析、数据挖掘。

朱争（1993−），男，四川攀枝花人，国防科技大学硕士生，主要研究方向为信息安全。

周斌（1971−），男，江西吉安人，博士，国防科技大学研究员、博士生导师，主要研究方向为数据挖掘、信息安全。

方滨兴（1960−），男，江西上饶人，博士，中国工程院院士，北京邮电大学教授、博士生导师，主要研究方向为计算机网络、信息安全、并行计算等。