一种基于 Sketch 的 Top-k 紧密中心性快速搜索算法

邵蓥侠[1)] 崔 斌[1)] 马 林[1)] 阴红志[2)]

[1)](北京大学信息科学技术学院高可信软件技术重点实验室 北京 100871)

[2)](昆士兰大学信息技术和电子工程学院 昆士兰 澳大利亚 4072)

摘 要 在大数据的时代背景下,由于网络数据(network data)能有效简洁地描述社交网络、电子商务、医疗记录、在线教育等多种应用中各类复杂关系,越来越受到工业界和学术界的关注.在社交网络分析任务中,一个基本操作是从网络中发现重要程度前 k 大的节点.紧密中心性(closeness centrality)是一种常见的节点重要性刻画指标,它用节点在网络中心的程度来反映节点的重要性.用紧密中心性衡量节点重要性进行节点搜索的问题称为 top-k 紧密中心性搜索问题.然而,传统的精确算法由于其多项式级别的复杂度无法高效地扩展到大规模的网络数据上.近来,研究人员提出了近似算法,通过牺牲结果精度来获得性能提升.通过分析发现,目前存在的近似算法虽然性能得到了有效提升,但是结果精度牺牲过大.为了解决这个问题,该文设计了一种新颖的近似算法,叫做基于 Sketch 的紧密中心性搜索算法.此近似算法应用了一个全新的计算方式,利用 Sketch 估计同一距离的邻居数目,然后得到近似的最短距离之和,最终得到各个节点的紧密中心性的估计值.此算法的时间复杂度为 $O(mtD_{\max})$,其中 t 是常数,$D_{\max}$ 是网络直径,m 是网络边数.根据实际社交网络的小世界现象的特性,此近似算法基本是个线性算法.最后,相比于目前存在的精确算法和近似算法,该文通过全面的实验验证了基于 Sketch 的紧密中心性搜索算法在时间性能和结果精度等两方面的优势.

关键词 紧密中心性;图算法;近似算法;图分析;社交网络

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2016.01965

A Fast Sketch-Based Approach of Top-k Closeness Centrality Search on Large Networks

SHAO Ying-Xia[1)] CUI Bin[1)] MA Lin[1)] YIN Hong-Zhi[2)]

[1)](*Key Laboratory of High Confidence Software Technologies (MOE), EECS, Peking University, Beijing* 100871)

[2)](*School of Information Technology and Electrical Engineering, The University of Queensland, Qld* 4072, *Australia*)

Abstract The advanced development of various technologies on social network, e-commerce and online education has contributed to an increasing amount of large-scale network data. Among all sorts of network analysis tasks, one basic task is to search important nodes in a network. Closeness centrality is one of the popular metrics which measure the importance of a node in a network. Based on the closeness centrality, the basic task is called top-k closeness centrality search. However, the existing exact approaches cannot process large-scale networks because of their polynomial time complexity. Recently, some approximation algorithms are proposed, which achieve high performance by sacrificing the precision of results. But according to our study, we find that the loss of the precision of results is too much. To improve the precision of results while maintaining the high performance, in this paper, we propose a Sketch-based approximation

收稿日期:2015-10-18;在线出版日期:2016-02-24.本课题得到国家自然科学基金(61272155,61572039)、国家"九七三"重点基础研究发展规划项目基金(2014CB340405)和深圳政府研究项目(JCYJ20151014093505032)资助.**邵蓥侠**,男,1988 年生,博士研究生,主要研究方向为大规模图数据分析.E-mail:simon0227@pku.edu.cn.**崔 斌**,男,1975 年生,博士,研究员,中国计算机学会(CCF)杰出会员,主要研究领域为数据库、社交数据分析、大规模图挖掘.**马 林**,男,1993 年生,本科生,研究方向为大规模图数据分析.**阴红志**,男,1984 年生,博士,主要研究方向为推荐系统.

algorithm for fast searching top-k closeness centrality in a large-scale network. The new algorithm is developed based on a new computation method which calculates the centrality by estimating the number of nodes within a certain distance by a data structure called FM-Sketch. The new algorithm has time complexity $O(mtD_{\max})$, where t is a constant, $D_{\max}$ is the diameter of a network and m is the number of edges in a network. With the small-world phenomenon assumption, the Sketch-based algorithm is a linear algorithm. Finally, we compare our Sketch-based algorithm with the state-of-the-art exact and approximation algorithms through extensive experiments. The results demonstrate the advantages of the new solution.

Keywords closeness centrality; graph algorithm; approximation algorithm; graph analysis; social network

1 引 言

近年来,随着万维网、Web2.0、移动网络、社交媒体以及电子商务等技术的迅速发展,大规模的网络数据已经无处不在.截至 2014 年 2 月,Facebook 拥有全球 12 亿用户,其中仅好友关系已多达 2016 亿条①.同时,根据 CNNIC 的统计②表明,截至 2013 年 12 月,国内社交网站用户规模达 2.78 亿.与此同时,社交网络分析(Social Network Analysis,SNA)作为一种应用性极强的社会学研究方法,越来越受大家关注.因为 SNA 不仅能给公司和企业带来巨大的正效益,而且能协助政府、公安机关进行有效的公共安全监控和预警,比如利用多种类型的社会关系挖掘网络中的犯罪团伙并进行预警.

社交网络分析任务中一个重要的研究问题是快速有效地发现网络中的重要节点[1-6].例如,在微博上发现有影响力的用户[1,7],并确定信息传播过程中的关键点[2]及在现实的生活社区中发现流行的商店[8]等.在网络分析任务中,一个网络节点的重要性通过中心性(Centrality)进行测量.经过研究人员多年的研究,目前已经提出了多种中心性的定义[9],包括基于信息流的 PageRank、HITS、中介(Betweenness)中心性、紧密(Closeness)中心性、基于度(Degree)的中心性等.每一种中心性都从不同角度刻画了一个节点在网络中的重要程度,适用于不同的应用场景.

本文重点研究的是基于紧密中心性(Closeness Centrality)的中心性测度.紧密中心性可以描述为一个节点到网络中各个节点的平均最短路距离的倒数.它是一个基于节点的单源最短路距离分布的衡量指标,形象地描述了节点在网络中所处位置的重要程度.紧密中心性越大的节点到其他节点的平均最短路径距离越小,处于网络的越中央;能更迅速地将信息传递给其他节点,同时也能更快速地接收其他节点的信息.在实际的社交网络中,如新浪微博、人人网、豆瓣等,一个紧密中心性大的用户的状态、日志等更新相对于其他用户的内容能更迅速地传播到用户所在的社区.

关于紧密中心性的一个重要的网络分析任务称为 top-k 紧密中心性节点搜索问题.该问题的核心是找到一个网络中紧密中心性前 k 大的几个节点.解决问题的关键步骤就是快速计算网络中每个节点的紧密中心性.

然而,在大数据时代背景下,上述分析任务面临的网络规模不断膨胀[10-12].尽管紧密中心性能有效地刻画网络节点的重要性,但是经典的计算方法已经无法高效地处理大规模的网络数据.朴素的算法需要穷举网络中任意点对的最短距离才能计算网络中所有节点的紧密中心性,这必然引入 $O(n^2)$ 的空间复杂度和 $O(n^2m)$ 的时间复杂度.如果处理一张含有百万(10^6)节点的网络数据就需要 1 太(10^{12})左右的空间和时间开销.最近,Olsen 等人[13]通过优化计算顺序,估算节点紧密中心性的上界值并利用该值过滤不合法的顶点的方式,减少冗余计算,最终提出了一种高效的精确算法.相对于朴素算法,Olsen 等人设计的精确算法性获得了很大的性能提升.但是在处理大规模网络数据时,此精确算法仍然表现出性能低下,主要是因为平均情况下它的计算复杂度仍然与网络节点数目成多项式的关系.

考虑到实际应用中,用户关注的是那些紧密中

① Facebook statistics. http://highscalability.com/blog/2014/2/14/stuff-the-internet-says-on-scalability-for-february-14th-201.html

② 中国互联网络发展状况统计报告. http://www.cnnic.net.cn/hlwfzyj/hlwxzbg/hlwtjbg/201403/P020140305346585959798.pdf

性大的节点，并非具体的紧密中心性的数值. 针对 top-k 紧密中心性这个分析任务，不少研究人员开始重点研究近似算法，通过快速估算节点的紧密中心性，提升算法性能. Eppstein 和 Wang[14] 通过随机选取一定数目的节点作为种子，利用各个节点到种子节点的最短路径估计节点的紧密中心性. 虽然此随机算法能高效地处理大规模的网络，但是结果的精度不是非常理想.

综合考虑，本文主要研究在处理大规模网络的背景下的紧密中心性节点搜索问题，即在给定网络 G 和阈值 k 的条件下，返回紧密中心性最大的前 k 个节点(详细定义见 2.2 节). 通过上述分析可知，目前的精确算法在大规模网络下性能表现差，而近似算法则表现出结果精度不理想.

本文作者重新分析了紧密中心性的计算结构，提出了一种基于 Sketch 的紧密中心性快速搜索近似算法. 此算法在处理大规模网络数据时，既能获得高效的性能，同时具有很高的结果精度. 在设计算法的过程中，笔者首先重新分析了紧密中心性的计算结构，提出了基于统计最短路分布的紧密中心性计算方法；然后，利用 FM-Sketch 数据结构对最短路分布进行估计，把多项式的算法空间和时间复杂度降到了线性的复杂度，从而保证算法性能.

另外，笔者还讨论了算法的结果精度与利用 FM-Sketch 对最短路分布估计的精度的关系，并提出了启发式规则进一步优化精度.

最后，通过全面的实验对比，笔者发现基于 Sketch 的算法性能比精确算法提升了 2 个数量级，比目前存在的近似算法快 2～3 倍左右；在精度方面，不同的(真实和模拟)网络数据上，新的近似算法比现存的近似算法高 10%～60%.

在本文第 2 节，笔者详细介绍紧密中心性、top-k 紧密中心性节点搜索问题、FM-Sketch 等基本概念，同时分别介绍现阶段效果最好的一种精确算法和近似算法；在第 3 节，笔者详细介绍基于 Sketch 的紧密中心性快速搜索近似算法；在第 4 节中通过全面的实验对比说明新方法的优势；在第 5 节中，笔者将详细讨论关于节点中心性计算的相关工作；最后，在第 6 节笔者对本文进行总结.

2　Top-k 紧密中心性搜索的基本概念

本节正式介绍紧密中心性的概念和基于紧密中心性的节点搜索问题的定义. 其次，笔者分别介绍一种关于紧密中心性搜索问题现阶段性能最好的精确算法和近似算法. 最后，笔者还简单介绍 FM-Sketch 的核心思想和功能.

2.1　紧密中心性定义

本文用 $G(V,E)$ 表示一张网络，其中 V 是网络 G 的顶点集合，E 是网络 G 的边集合. 在社交网络上的节点中心性分析任务中，网络 G 是一张无权网络，即边和节点不带权值. 网络中的节点用小写字母 v,u 表示. 网络中节点数和边数分别记为 n 和 m. 另外，$d(u,v)$ 表示网络上两节点 u,v 之间的最短路距离；$c(v)$ 表示顶点 v 的紧密中心性. 表 1 总结了本文将要频繁使用的符号.

表 1　本文常用符号汇总

符号	含义
$G(V,E)$	一张无权的网络
n,m	网络上的节点数和边数
u,v	网络上的节点
$c(v)$	节点 v 的紧密中心性
$s(v)$	节点 v 到其他节点的最短距离之和
$d(u,v)$	节点 u 和 v 之间的最短路距离
$D_{\max}$	网络的直径
$N_d(v)$	与节点 v 最短路距离为 d 的节点数目
$N_{\leq d}(v)$	与节点 v 最短路距离不超过 d 的节点数目
$NV_{\leq d}(u)$	与节点 u 最短路距离不超过 d 的节点的集合
$FM_{\leq d}(v)$	与节点 v 最短路距离不超过 d 的 FM-Sketch
$\lvert FM_{\leq d}(v)\rvert$	$FM_{\leq d}(v)$ 对应的估计节点数目

之前，笔者已经简单说明紧密中心性与节点的平均最短距离之间的关系. 下面定义 1 给出了紧密中心性的形式化定义.

定义 1(紧密中心性).　在一个网络 G 中，节点 v 的紧密中心性 $c(v)$ 是它到其他节点的平均最短路距离的倒数，形式化表示为

$$c(v)=\frac{n-1}{\sum\limits_{\{v'\in V\}}d(v,v')} \tag{1}$$

其中，n 表示网络中节点数目；$d(v,v')$ 表示节点 v 和 v' 之间的最短路距离.

利用定义 1 计算节点的紧密中心性，需要图 G 是连通图的假设，否则会出现 $d(v,v')=\infty$ 的情况. 由于现实世界中的网络存在多个连通块，使得定义 1 无法有效计算节点在全网络中的紧密中心性. 这里笔者引入一种更加全面通用的紧密中心性计算方式. 在任意给定的一个网络 G 中，节点 v 的紧密中心性表示为

$$c(v)=\frac{(\lvert V_v\rvert-1)^2}{(n-1)\sum\limits_{v'\in V_v}d(v,v')} \tag{2}$$

其中，$|V_v|$ 表示包含节点 v 的连通块中顶点数

目.式(2)可以理解为每个节点先计算其在连通块内的紧密中心性 $\frac{(|V_v|-1)}{\sum_{v'\in V_v} d(v,v')}$,然后通过乘以归一化系数 $\frac{|V_v|-1}{n-1}$ 得到全局的紧密中心性.这样使得每个节点的紧密中心性在全局满足偏序关系,具有可比性.

2.2 Top-k 紧密中心性搜索问题定义

本文研究如何在一张大规模网络中快速找到紧密中心性前 k 大的节点,简称为 top-k 紧密中心性节点搜索问题.下面是此问题的具体定义.

定义 2(top-k 紧密中心性节点搜索). 给定参数 k,在网络 $G(V,E)$ 上寻找 k 个节点,使得这 k 个节点的紧密中心性在网络 G 中是前 k 大.

考虑到问题的通用性,本文在解决 top-k 紧密中心性搜索问题时,利用式(2)计算节点的紧密中心性.

2.3 紧密中心性搜索的精确算法 Δ-pfs

精确算法是指通过准确计算节点的紧密中心性的方式,求得精确的前 k 大节点.目前最新且性能最好的精确算法是 Olsen 等人[13]在 2014 年的数据库国际会议 ICDE 上提出的 Δ-pfs 算法.

经典的精确算法需要利用单源最短路算法给每一个节点计算准确的紧密中心性,然后根据紧密中心性取值排序,得到前 k 大的节点.然而,经典算法中存在大量的冗余计算,即不同节点在计算单源最短路时,存在重复计算的最短路.为了减少冗余计算,合理利用已经计算得到的最短路,Olsen 等人提出了 Δ-pfs 算法.

Δ-pfs 算法优化的核心思想是在计算某一个顶点的紧密中心性时重复利用其他顶点计算紧密中心性的中间结果.图 1 具体展示了一个冗余计算共享的例子.图 1(a)展示了在一张包含 6 个顶点和 7 条边的有向图上计算顶点 b 的紧密中心性的过程.其中,$d(b,f)=d(b,g)=1$, $d(b,b)=0$, $d(b,h)=2$,于是 $c(b)=\frac{(|V_b|-1)^2}{(n-1)\sum_{v'\in V_b} d(v,v')}=\frac{(4-1)^2}{(6-1)(0+1+1+2)}=\frac{9}{20}$.

在图 1 基础上,计算顶点 a 的紧密中心性,朴素的做法需要全部遍历图中的 7 条边,计算顶点 a 与其他节点对之间的最短距离之和.Olsen 等人重新设计算法,通过利用先前的搜索避免重复计算,有效提升算法性能.首先,在计算顶点 b 的紧密中心性

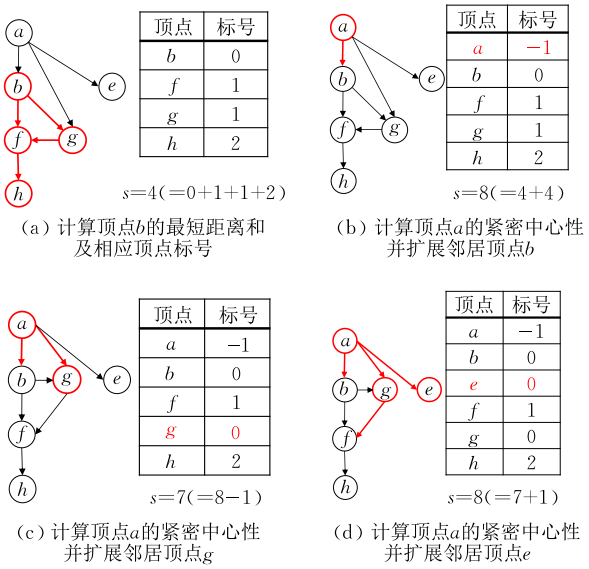


图 1 Δ-pfs 算法冗余计算共享示例

时,根据计算的点对之间的最短距离 $d(b,v)$,把每个顶点 v 标号为 $d(b,v)$,然后把其他顶点 a 能达到顶点标记为 $d(b,v)-w(a,b)$,其中 $w(a,b)$ 为边 (a,b) 的权重,最后在计算过程中,新标号与旧标号一致就可以跳过重复计算.

通过计算顶点 b 的紧密中心性,顶点 b,f,g,h 的标号如图 1(a)中的表格所示.现在,当计算顶点 a 的紧密中心性时,如图 1(b)所示,首先顶点 a 的标号为 $d(a,a)-w(a,b)=-1$,然后根据顶点 a 的出边,搜索到顶点 b,计算得到顶点 b 的新标号为 $d(a,b)-w(a,b)=0$,与其旧标号一致,说明从顶点 b 出发的顶点都被计算过,且标号不用更新,因此可以跳过对顶点 b 的继续搜索,避免了不必要的计算.而当从 a 出发计算顶点 g 时(图 1(c)),顶点 g 的新标号为 $d(a,g)-w(a,b)=0$ 与其旧标号 1 不同,因此继续扩展顶点 g,计算可得顶点 f 的新旧标号一致,停止扩展.而顶点 e 则由于没有旧标号(图 1(d)),同样需要进一步的扩展更新.

根据标号变化,可以相对快速地计算得到新的顶点的最短路距离之和,用变量 s 维护该值.当计算顶点 b 以后,s 的值为 4.在计算顶点 a 时,由于起点的标号变为 -1,这样从 b 出发的可达的顶点的最短距离每个增加 1,因此 s 增加 4×1.由于顶点 g 的标号从 1 变为 0,这样 s 减少 1 以及顶点 e 是新引入的 s 增加 $d(a,e)=1$,最终得到最新的 s 为 8.于是 $c(a)=\frac{(|V_a|-1)^2}{(n-1)\sum_{v'\in V_a} d(v,v')}=\frac{(6-1)^2}{(6-1)8}=\frac{5}{8}$.

为了有效降低计算的冗余度，作者进一步利用代价模型构建一张代价网络. 在此网络上，利用最小生成树算法选择合适的初始节点集合，并按一定的顺序进行单源最短路的计算，使得后续的节点计算能像之前分析的一样有效利用已经处理过的节点的中间结果，减少冗余计算. 其次，Δ-pfs 算法根据已知的计算结果，为每个节点估计紧密中心性的上界，利用估计的上界，及时过滤那些不可能属于前 k 大的节点避免无意义的计算. 通过上述优化，相对于传统算法，Δ-pfs 算法性能得到了有效的提升.

然而，Δ-pfs 算法的平均时间和空间复杂度很难讨论清楚，实际优化性能随着数据的不同而波动很大. 最坏情况下它的时间和空间复杂度上仍然是多项式的，无法高效处理现实世界中的大规模网络数据.

2.4　紧密中心性搜索的近似算法 RAND

为了能在大规模网络上高效计算紧密中心性搜索问题，研究人员通过设计近似算法对紧密中心性进行估计，进而获得近似的 top-k 紧密中心性节点.

早在 2004 年，Eppstein 提出了一种快速估计节点紧密中心性的随机算法，记为 RAND-算法[14]. 精确算法中一个节点的紧密中心性由节点到所有其余节点的最短路之和决定；而 RAND-算法的思想是利用随机抽样的方法选出 s 个(可重复的)节点作为种子，然后一个节点的紧密中心性由节点到这 s 个节点的最短距离之和进行估计. 这样就把计算 n 次单源最短路降到了计算 s 次单源最短路. RAND-算法的基本步骤如下：

(1) 令 s 为算法迭代轮数；

(2) 在第 i 轮迭代中，从给定网络 G 中随机选择节点 v_i，并计算以 v_i 为源点的单源最短路；

(3) 网络 G 中节点 u 的紧密中心性按如下公式进行估计：

$$c(u)=\frac{1}{\sum_{i=1}^{s}\frac{nd(u,v_i)}{s(n-1)}}.$$

作者证明当 RAND-算法抽样 $O\left(\frac{\log n}{\varepsilon^2}\right)$ 个种子时，节点的平均最短路误差控制在 εD，其中 D 是网络的直径. 这样 RAND-算法是一个 $(1+\varepsilon)$ 近似的算法. 然而在实际应用中，为了得到精度不错的结果，s 往往会比较大；或者当 s 较小的时候，RAND-算法的结果精度较低.

2.5　FM-Sketch 介绍

接下来，笔者介绍后续在设计本文算法时需要用到的一个数据结构 FM-Sketch[15]. 它是一种高效且通用的估计大规模集合基数(cardinality)的方法. 此方法基于元素哈希值的比特位(bit)分布特性，能仅利用几百字节的空间通过扫描一遍集合快速估计集合内不同元素的个数.

FM-Sketch 的计算过程是：对于集合中的每一个元素 x，通过一个哈希函数 h 将其映射到一个固定长度(t 位)的二进制数 $h(x)$；然后利用生成的 $h(x)$ 构造如下一个 t 位二进制的 bitmap. 首先把 bitmap 每一位都置为 0；然后计算集合中每个元素的 $h(x)$ 中最低位 1 的位置，并把 bitmap 中相应位置设为 1；这里，多个元素的 $h(x)$ 对一个 bitmap 的操作的结果等价于每个 $h(x)$ 对应的 bitmap 的逻辑运算“或”的结果. 最后，当集合中所有元素处理完以后，计算 bitmap 中最低位 0 的位置，记为 r，原集合中不同元素的数目估计值为 1.3×2^r，其中常数 1.3 是一个基于统计的魔法数字，用于修正估计结果. 在实际运用中，为了提高精度，通常使用多个哈希函数，并取所有 r 的平均值进行估计. FM-Sketch 的原理和相关证明可以参考文献[15].

下面再通过一个具体例子说明如何进行估计. 假设集合中有 4 个元素 x_1，x_2，x_3 和 x_4，通过 3 个哈希函数 $h_1(x)$，$h_2(x)$ 和 $h_3(x)$，生成 3 个 bitmap B_1，B_2，B_3. 每一个哈希函数都将任意一个元素映射到一个 $t=4$ 位的二进制. 其中表 2 展示了 $h_1(x)$ 的计算结果；表 3 列出了 3 个哈希函数作用下生成的 3 个 bitmap. 以 $h_1(x)$ 为例，得到对应的 bitmap B_1 的二进制值为 $(1011)_2$，这样 $r_1=2$. 同理可得，$r_2=2$，$r_3=1$. 于是，$r=\frac{2+2+1}{3}=1.67$，估计的集合大小约为 $1.3\times 2^{1.67}=4.14$.

表 2　4 个元素在哈希函数 h_1 下的哈希值
(表中一行左边第一位表示二进制的最低位)

$h_1(x_1)$	$h_1(x_2)$	$h_1(x_3)$	$h_1(x_4)$
1	0	0	1
0	0	0	0
1	1	0	0
0	1	1	1

表 3　3 个哈希函数对应的 bitmap 值

B_1	B_2	B_3
1	0	0
0	0	1
1	1	0
1	1	1

除了利用多个哈希函数求平均提高估算精度的方法以外，通过对同一个哈希函数的哈希值进行分组处理，然后根据不同分组的估算结果进行综合得

到最终的估算结果.作者讨论了在 t 固定的条件下,随着组数 m 的增大,FM-Sketch 的估算精度会提高.

这里具体的估算方法如下:假设 n 个不同元素在同一个哈希函数下计算的哈希值,被划分成 m 组 FM-Sketch,那么 n 的估算公式如下:

$$n \approx m \times 2^r \times 1.3,$$

其中 r 是指 m 组 bitmap 中最低位 0 的位置的平均值.

3 基于 Sketch 的紧密中心性搜索算法

本节详细介绍基于 Sketch 的 top-k 紧密中心性搜索算法.首先,笔者给出了一种基于计数的紧密中心性计算方式,并结合此新的计算方式,设计了全新的紧密中心性搜索算法;然后,利用 FM-Sketch 数据结构进行优化,使得算法能够快速估计各个节点的紧密中心性.最后,笔者讨论了近似算法结果精度,利用启发式规则进一步优化算法精度.

3.1 基于计数的紧密中心性计算

根据式(2)可知,紧密中心性的计算核心是统计一个节点 u 到其他节点的最短路距离之和,记为 $s(u)$.如果知道一个节点 u 的所有最短路按其距离长度的分布,那么节点 u 的所有最短路距离之和可以表示为

$$s(u) = \sum_{v \in V} d(u,v) = \sum_{d=1}^{D_{\max}} d \times N_d(u) \qquad (3)$$

其中,$D_{\max}$ 表示网络 G 上任意点对中最长的最短路距离,即为网络 G 的直径;$N_d(u)$ 表示与节点 u 最短路距离为 d 的节点数目.

根据式(3),紧密中心性的计算转化为一定长度的最短路径数目的统计.用 $NV_{\le d}(u)$ 表示与节点 u 最短路距离不超过 d 的节点的集合,即 $NV_{\le d}(u) = \{v \mid d(u,v) \le d\}$.于是,$N_d(u) = |NV_{\le d}(u)| - |NV_{\le d-1}(u)|$.

结合社交网络应用中重要节点发现的特征,本文考虑的网络不涉及节点和边的权值,即所有边的长度均视为 1,于是 $NV_{\le d}(u)$ 满足如下递推性质.

性质 1. 节点 u 在距离 d 以内可达的节点集合是该节点的所有邻居节点在距离 $d-1$ 以内可达的节点集合的并集,即

$$NV_{\le d}(u) = \bigcup_{(u,v) \in E} NV_{\le d-1}(v).$$

利用式(2)、(3)和性质 1,可以在网络 $G(V,E)$ 上搜索 top-k 紧密中心性节点.基本思想是通过逐个扫描网络 G 上的连通块,针对每一个连通块 CC_v,利用计数的方式计算每个连通块内节点的最短路距离之和.最终,利用性质 1,能够在 $D_{\max}$ 次循环求得连通块内所有节点的最短路之和.

算法 1 详细展示了上述过程.对于给定的网络 $G=(V,E)$,算法枚举遍历网络中的顶点 v(第 1 行),如果该顶点 v 没有被访问过,即其不属于任何被处理过的连通块中,算法 1 利用 BFS 搜索策略找到顶点 v 所在的连通块 CC_v,并把连通块内的所有节点标记为已访问;否则跳过顶点 v 继续处理下一个顶点(第 1～4 行).对于找到的 CC_v,首先利用抽样算法估算 CC_v 的直径 $D_{\max}$(第 5 行),然后 $D_{\max}$ 次迭代利用式(2)、(3)计算连通块内每个顶点的最短距离之和(第 9～17 行).最后,根据顶点紧密中心性的计算公式计算相应节点的紧密中心性,并更新结果集合(第 18 行).

算法 1. 基于计数的 top-k 紧密中心性搜索算法.

输入:网络 $G=(V,E)$,k

输出:紧密中心性 top-k 的节点编号

1. $ansSet \leftarrow \varnothing$;
2. for $v \in V$ do
3. if v 没有被访问过 then
4. 计算节点 v 所在的连通块 CC_v,并把连通块内的所有节点标记为访问过;
5. $D_{\max} \leftarrow CC_v$ 的直径;
6. for $w \in CC_v$ do
7. $N_{\le 0}(w) \leftarrow \{w\}$;
8. end for
9. for $d \leftarrow 1$ to $D_{\max}$ do
10. for $w \in CC_v$ do
11. $N_{\le d}(w) \leftarrow \varnothing$;
12. for $u \in Neighbor(w)$ do
13. 计算 $N_{\le d}(w)$ 与 $N_{\le d-1}(u)$ 的并集;
14. end for
15. 更新 $s(w)$,即 $s(w) \leftarrow s(w) + d \times (|N_{\le d}(w)| - |N_{\le d-1}(w)|)$;
16. end for
17. end for
18. for $w \in CC_v$ do
19. 计算 $c(w)$,即 $c(w) \leftarrow \frac{(|CC_v|-1)^2}{(n-1)s(w)}$;
20. 根据 top-k 限制,利用 $c(w)$ 更新 $ansSet$;
21. end for
22. end if
23. end for
24. return $ansSet$;

接下来，通过一个具体例子详细介绍基于计数的紧密中心性计算过程. 给定网络 G 如图 2 所示和 $k=2$. 网络 G 的直径 $D_{\max}=3$，只有一个连通块. 表 4 列出了不同迭代轮次内每个顶点的邻居集合.

表 4　不同迭代轮次各顶点的邻居节点集合

d	$NV_{\le d}(a)$	$NV_{\le d}(b)$	$NV_{\le d}(c)$	$NV_{\le d}(d)$	$NV_{\le d}(e)$	$NV_{\le d}(f)$
0	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{e\}$	$\{f\}$
1	$\{a,b,e\}$	$\{a,b,c,f\}$	$\{c,b,d,f\}$	$\{d,c\}$	$\{e,a,f\}$	$\{f,b,c,e\}$
2	$\{a,b,e,f,c\}$	$\{a,b,c,f,e,d\}$	$\{c,b,d,f,e,a\}$	$\{d,c,b,f\}$	$\{e,a,f,b,c\}$	$\{f,b,c,e,a,d\}$
3	$\{a,b,e,f,c,d\}$	$\{a,b,c,f,e,d\}$	$\{c,b,d,f,e,a\}$	$\{d,c,b,f,a,e\}$	$\{e,a,f,b,c,d\}$	$\{f,b,c,e,a,d\}$

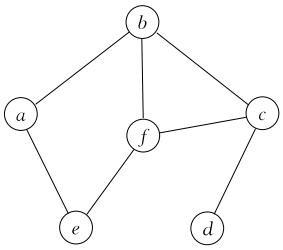


图 2　1 个包含有 6 个顶点和 7 条边的无向无权图

结合算法 1，可知基于计数的紧密中心性计算一共包含 3 轮迭代. 在初始化阶段(第 0 轮迭代)，每个顶点的邻居集合即为它本身，如表 4 第 1 行所示. 随着迭代的进行，每个顶点的集合通过其邻居顶点的上一轮迭代的邻居集合合并得到最新的集合元素. 以顶点 a 为例，在第 2 轮结束后，它的最短路距离不超过 2 的邻居集合，

$$\begin{aligned} NV_{\le 2}(a) &= NV_{\le 1}(a) \cup NV_{\le 1}(e) \cup NV_{\le 1}(b) \\ &= \{a,b,e\} \cup \{a,b,c,f\} \cup \{e,a,f,b,c\} \\ &= \{a,b,e,f,c\}. \end{aligned}$$

直到 3 轮迭代结束，由于本样例是个连通图，因此每个顶点的最短路距离不超过 3 的邻居集合都是 $\{d,c,b,f,a,e\}$.

复杂度分析. 算法 1 中，每个节点需要计算关于一定距离内可达节点的集合运算(代码第 13 行)，因此每个节点需要维护上述邻居集合 $NV_{\le d}(u)$，如表 4 所示. 令该集合的平均空间大小为 M_{cost}，于是算法 1 的空间复杂度为 $O(nM_{\text{cost}})$. 同样，时间复杂度为 $O(mD_{\max}T_{\text{cost}})$，其中 T_{cost} 是指计算一次邻居集合并的平均时间代价.

现实世界中的社交网络往往一般具有小世界现象，一个节点与其他所有节点的平均最短距离比较小，接近于常数[16]，即 $D_{\max}$ 相对于网络规模可视为常数. 因此算法 1 的时间和空间代价基本由邻居集合的运算和大小决定. 然而，在计算过程中，随着 d 的增大，当 d 接近 $D_{\max}$ 时，邻居集合 $NV_{\le d}(u)$ 基本接近全图顶点，使得此精确算法仍然是关于网络规模 n 或 m 的多项式算法，很难扩展到大规模的网络图上.

3.2　基于 Sketch 的近似算法

基于计数的紧密中心性计算的瓶颈在于维护 $NV_{\le d}(u)$ 和计算相应的集合并. 即使利用位图(bitmap)为每个节点 u 维护精确的 $NV_{\le d}(u)$，只能降常数级别的复杂度，算法总体上仍然需要多项式级别的空间和时间复杂度. 笔者利用 Sketch 的方法，为每个节点维护一个近似的 $NV_{\le d}(u)$，从而估算节点的紧密中心性. 此法称为基于 Sketch 的紧密中心性搜索算法.

首先，维护 $NV_{\le d}(u)$ 的过程可以抽象为“从节点 u 开始向外扩展 d 步，生成一组节点的访问序列，$NV_{\le d}(u)$ 则为此序列中互不相同的节点的集合”. 集合内元素的数目用 $|NV_{\le d}(u)|$ 表示. 换而言之，维护 $NV_{\le d}(u)$ 的过程就是随着节点往外扩展，算法需要动态维护集合 $NV_{\le d}(u)$ 中不同元素的个数. 在 2.5 节中介绍的 FM-Sketch 恰好是一个用于估计此问题的很好的数据结构. 这里笔者引入 FM-Sketch 对 $|NV_{\le d}(u)|$ 进行估计.

根据之前关于 FM-Sketch 的介绍，可以知道 FM-Sketch 具有如下几点性质：

(1) FM-Sketch 的值与其加入元素的顺序无关；

(2) FM-Sketch 是统计一个集合内不同元素的个数，能自动过滤掉重复元素；

(3) 集合并运算可以简单地转化为基于 FM-Sketch 的逻辑“或”运算. 例如，估计两个独立的集合 A 和 B 的并集 C 中包含的不同元素的个数可以通过 A 和 B 的 FM-Sketch 的“或”运算得到，即 $FM(C)=FM(A)|FM(B)$.

(4) 具有包含关系的两个集合的差集大小同样可以通过相应 FM-Sketch 的差值得到. 例如，给定集合 A 和 B，且 $B\subseteq A$，那么 $|A-B|=|FM(A)|-|FM(B)|$.

结合 FM-Sketch 的上述特性，top-k 紧密中心性搜索的算法 1 的近似版本如算法 2 所示. 此近似算法的空间复杂度为 $O(tn)$，其中 t 是维护 FM-

Sketch 所需的位数;时间复杂度为 $O(mtD_{\max})$. 从上面我们可以发现,近似算法由于引入了 FM-Sketch 数据结构,虽然牺牲了一定的结果精度,但是算法的空间复杂度变为线性,时间复杂度仅与网络的边数和直径相关. 由于实际网络一般比较稀疏和具有小世界现象,此近似算法时间复杂度基本是关于网络中边的数目的线性时间复杂度.

算法 2. 基于 Sketch 的 top-k 紧密中心性搜索算法.

输入:网络 $G=(V,E)$,k

输出:紧密中心性 top-k 的节点编号

1. $ansSet \leftarrow \varnothing$;
2. for $v \in V$ do
3. if v 没有被访问过 then
4. 计算节点 v 所在的连通块 CC_v,并把连通块内的所有节点标记为访问过;
5. $D_{\max} \leftarrow CC_v$ 的直径;
6. for $w \in CC_v$ do
7. 初始化节点 w 的 FM-Sketch,即 $FM_0(w)$;
8. end for
9. for $d \leftarrow 1$ to $D_{\max}$ do
10. for $w \in CC_v$ do
11. $FM_{\le d}(w) \leftarrow 0$
12. for $u \in Neighbor(w)$ do
13. $FM_{\le d}(w) = FM_{\le d}(w) \mid FM_{\le d-1}(u)$;
14. end for
15. 更新 $s(w)$,即 $s(w) \leftarrow s(w) + d \times (|N_{\le d}(w)| - |N_{\le d-1}(w)|)$;
16. end for
17. end for
18. for $w \in CC_v$ do
19. 计算 $c(w)$,即 $c(w) \leftarrow \frac{(|CC_v|-1)^2}{(n-1)s(w)}$;
20. 根据 top-k 限制,利用 $c(w)$ 更新 $ansSet$;
21. end for
22. end if
23. end for
24. return $ansSet$;

3.3 近似算法精度讨论及优化

接下来笔者讨论算法结果精度与 FM-Sketch 精度的关系. 直观的感觉是如果 FM-Sketch 的估计结果越精确,那么算法 2 产生的结果精度也就越高. 定理 1 给出了详细的理论分析.

定理 1. 如果一个连通网络的直径为 D,那么算法 2 得到的每一个节点的最短路距离和 $s(u)$ 的方差 δ_s 为

$$\delta_s = D^2 \times \delta_{FM_D} + \sum_{d=1}^{D-1} \delta_{FM_d},$$

其中,δ_{FM_d} 表示距离 d 以内的利用 FM-Sketch 估计的节点数目的方差.

证明. 根据算法 2 中第 10 行代码可知

$$s(u) = \sum_{d=1}^{D} d \times (|FM_{\le d}(u)| - |FM_{\le d-1}(u)|)$$

$$= D \times |FM_{\le D}(u)| - \sum_{d=1}^{D-1} |FM_{\le d}(u)|.$$

由于 FM-Sketch 的方差仅与其使用的哈希函数相关,不同距离的 FM-Sketch 估计相互独立. 于是可以得到

$$\delta_s = D^2 \times \delta_{FM_D} + \sum_{d=1}^{D-1} \delta_{FM_d}. \qquad \text{证毕.}$$

根据定理 1,为了提高算法 2 的精度,原则是提升节点数目估计的精度. 这里我们介绍一种启发式规则,能有效提升算法精度. 在一定的内存开销下,即 t 固定,当估计的集合大小与对应的 FM-Sketch 能表示的最大数目接近时,估计精度越高.

然而,在 d 比较小的时候,如 d 为 1 或 2,由于访问的节点数目过小,FM 的估计精度很低. 考虑到 d 较小的时候,节点数目不多,直接精确计算节点数目的代价也很小. 这样笔者为算法 2 引入了如下启发式规则.

启发式规则. 当 $d=1,2$ 时,算法 2 精确计算节点数目;当 $d>2$ 时,算法 2 利用 FM-Sketch 估计相应节点数目.

从后续的实验可以看出,此启发式规则在处理的网络数据直径比较小的时候,能有效提升算法的精度.

4 实验与算法比较分析

本节通过实验测试,验证基于 Sketch 的紧密中心性节点搜索算法在大规模网络数据上的处理性能.

4.1 实验环境

本文实验运行在一台 Linux 服务器上,此服务器装有 2 个 Xeon(R) E5530@2.40 GHz 的 CPU,96 GB 内存以及 4 个 2 TB 硬盘. 实验中涉及的网络数据由两类组成:一是由开源软件 NetworkX① 生成的模拟网络数据;二是斯坦福大学的 SNAP 网站②下载得到的真实网络数据. 其中,生成模拟网络

① NetworkX. https://networkx.github.io/
② SNAP. http://snap.stanford.edu/data/index.html

数据的模型是 Holme-Kim 生长模型[17]. 此模型生成的图用 GHK(n,c)表示，n 是顶点数目，c 代表为每个新生成的顶点加入的边数. 表 5 列出了实验所使用的网络的具体信息.

表 5　网络元数据信息

网络类型	网络	n	m	D
模拟图	GHK(10k,20)	10k	399164	3
	GHK(20k,20)	20k	799172	3
	GHK(30k,20)	30k	1199164	3
	GHK(40k,20)	40k	1599172	3
	GHK(50k,20)	50k	1999160	3
真实图	ca-HepPh	12008	237010	13
	ca-CondMat	23133	186936	14
	email-Enron	36692	367662	11
	loc-brightkite	58228	428156	16
	loc-gowalla	196591	1900654	14
	com-youtube	1134890	2987624	20
	soc-LiveJournal1	4847571	68993773	16

精确算法，Δ-pfs 算法；效果最好的近似算法，RAND 算法，其中 $\varepsilon=0.1$；本文设计的 Sketch 算法；以及加入启发式规则优化的 Sketch 算法，记为 Sketch2hop 算法.

另外，Sketch 算法中使用的 FM-Sketch 的相关参数如下：bitmap 的比特位数设置为 24，即 $t=24$，这样基本能统计约 2^{24} 个不同元素的结合；使用一个哈希函数；哈希值被分成 128 组，即 $m=128$.

上述算法的性能主要从两方面进行比较：第一是算法运行时间；第二是算法的结果精度.

时间性能是算法重复运行 10 次的平均时间，精确到 0.01 s.

结果精度用准确率(precision)表示. 一组 top-k 结果的准确率按如下公式计算：

$$\text{precision@}k=\frac{|\{\text{算法输出结果}\}\cap\{\text{精确结果}\}|}{k}.$$

4.2　FM-Sketch 精度与算法精度关系

由定理 1 可知，本文算法的精度与 FM-Sketch 的精度有着紧密的联系，即 FM-Sketch 的估计精度越高，本文的 Sketch 算法的精度也越高. 笔者通过下面这组实验验证了上述观点.

由 2.5 节分析可知，FM-Sketch 的精度可以通过增大分组数 m 得到. 图 3 展示了在真实图 loc-gowalla 和 com-youtube 以及模拟图 GHK(10k,20)和 GHK(50k,20)上 m 从 8 变到 1024 过程中，Sketch 算法结果精度的变化. 从图上可以明显看出，Sketch 算法的精度随着分组数的变大(即 FM-Sketch 的估计精度提高)而变大. 例如，在真实数据图 com-youtube 上，当 $m=16$ 时，Sketch 算法精度是 64.08%，当 m 增大到 1024 时，Sketch 算法的结果精度约为 94.08%. 这里需要注意的一点是，在模拟图 GHK(10k,20)上，当 $m=32$ 时，Sketch 算法精度略差于 $m=16$ 时的精度，这是由于本文只用一个哈希函数以及数据特征的特殊性造成的.

4.3　算法时间性能比较

本节实验详细比较了 4 种算法的时间性能. 从实验结果可以看出，不管在真实图还是在模拟图上，Sketch 算法时间性能最好，比精确算法 Δ-pfs 快 2 个数量级，比 RAND 算法快 2～3 倍左右. 这个性能差距在大规模的网络数据上表现尤为明显. 表 6 列出了各个算法在不同类型图上的性能.

表 6　Sketch，Sketch2hop，RAND 和 Δ-pfs 这 4 种算法在不同数据集上的时间性能　(单位：s)

	GHK(10k,20)	GHK(20k,20)	GHK(30k,20)	GHK(40k,20)	GHK(50k,20)	ca-HepPh
Sketch	**0.38**	**0.86**	**1.41**	**1.95**	**2.53**	**0.47**
Sketch2hop	0.66	1.48	2.35	3.23	4.14	0.57
RAND	1.17	2.78	4.72	6.74	8.87	0.71
Δ-pfs	11.92	50.72	133.84	300.62	442.24	5.12
	ca-CondMat	email-Enron	loc-brightkite	loc-gowalla	com-youtube	soc-LiveJournal1
Sketch	**0.84**	**1.33**	**2.14**	**6.88**	**44.95**	**471.16**
Sketch2hop	0.89	1.61	2.31	10.06	61.53	640.76
RAND	1.19	1.72	2.97	13.50	83.39	1274.52
Δ-pfs	12.09	14.68	42.45	718.75	12208.43	>48 h

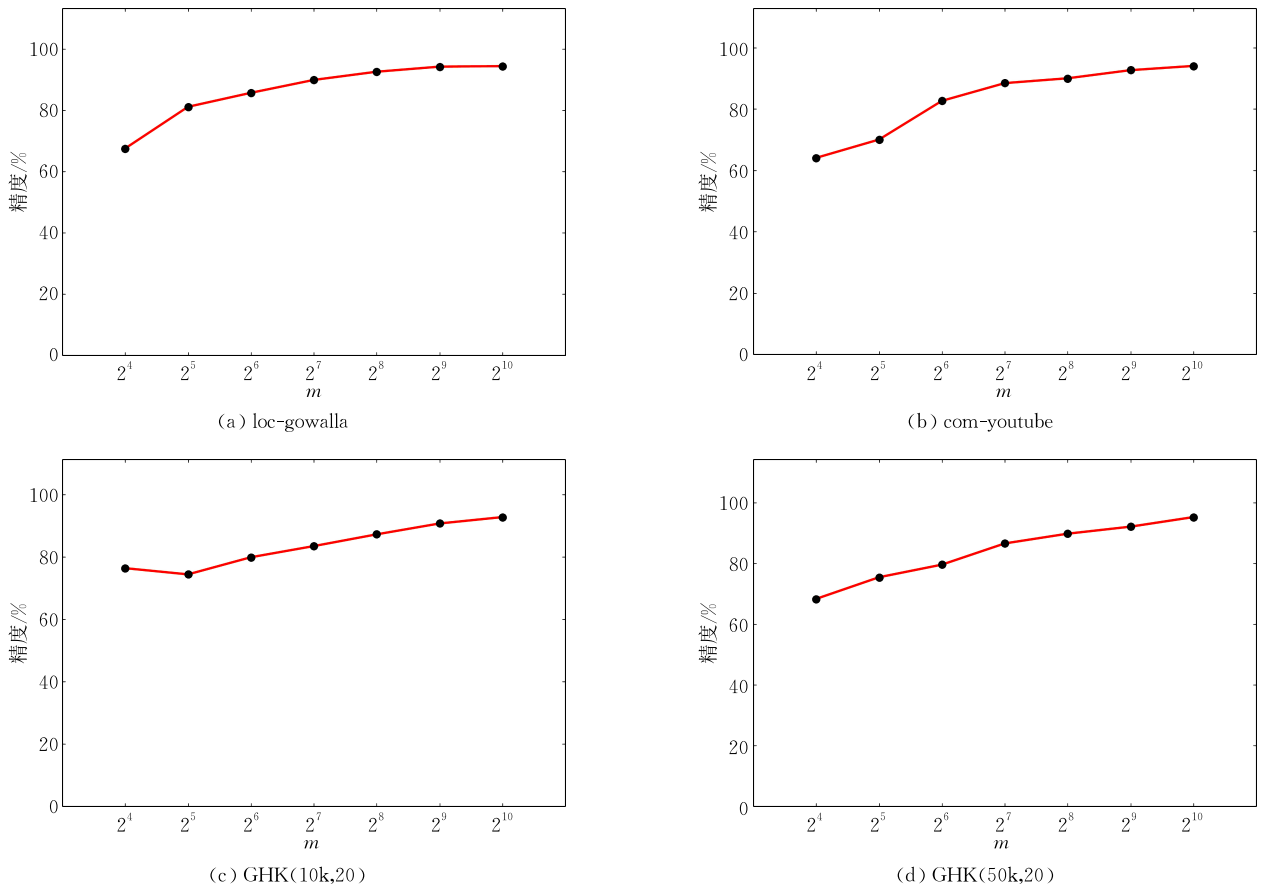


图 3 Sketch 算法精度(MAP@50)与 FM-Sketch 分组数(m)的变化关系

例如,在 com-youtube 数据上,求解 top-50 的紧密中心性搜索问题,Δ-pfs 耗时12 208 s;而 Sketch 算法仅需要 44 s. RAND 算法耗时接近 Sketch 算法的两倍,83 s. 在 LiveJournal 网络上,精确算法在 48 h 内仍然无法求得结果,Sketch 算法大约 10 min 左右返回结果. Sketch 算法的优秀的性能得益于其基于 FM-Sketch 的快速统计.

另外,从结果可以发现,Sketch2hop 算法时间性能虽然略差于 Sketch 算法,但是其时间性能仍优于 RAND 和 Δ-pfs 算法的时间性能. 这说明启发式规则对算法时间性能的影响不大,可以接受.

4.4 算法结果精度比较

本节实验对比了 3 种近似算法的结果精度,准确的结果由 Δ-pfs 算法生成. 由于 Livejournal 的精确结果无法在一定的时间内得到,本文没有比较该图上的结果精度.

实验过程中,每个算法分别计算了 top-1 至 top-50 的精度. 图 4 展示了 loc-gowalla, com-youtube, GHK(10k,20)和 GHK(50k,20)这 4 个数据集上精度随 k 的变化的情况. 从图上可以发现,基于 Sketch 的算法的精度在真实图和模拟图上均比 RAND 算法高,不过在不同类型图上提升效果不同. 在直径较大的真实社交网络中,Sketch 算法的精度能比 RAND 高 40%～60%;而在直径较小的图上优势略有下降,精度提升基本在 3%～8%.

另外,启发式规则在直径较小的网络上优化效果明显,Sketch2hop 算法的精度比 Sketch 算法高 10%左右;而在直径较大的网络上优化效果不明显. 从定理 1 可知,直径越大,误差越大,使得启发式规则的优化效果从整体上看就不明显.

其他数据集上的精度结果在不同的 k 值上的变化与上述 4 个数据集的情况类似,本文不一一展示结果.

表 7 列出了各个数据集上 top-1 至 top-50 的平均精度,即 MAP@50 指标. 从表中同样可以发现 Sketch 系列算法的精度比 RAND 算法高. 其中 Sketch2hop 算法在模拟图上精度可以达到 100%,即精确结果. 这是因为模拟图直径很小,从表 5 可知,直径均是 3,这样结合启发式规则,精确统计二跳以内邻居数目,从性能角度分析,Sketch2hop 算法与精确算法几乎等价.

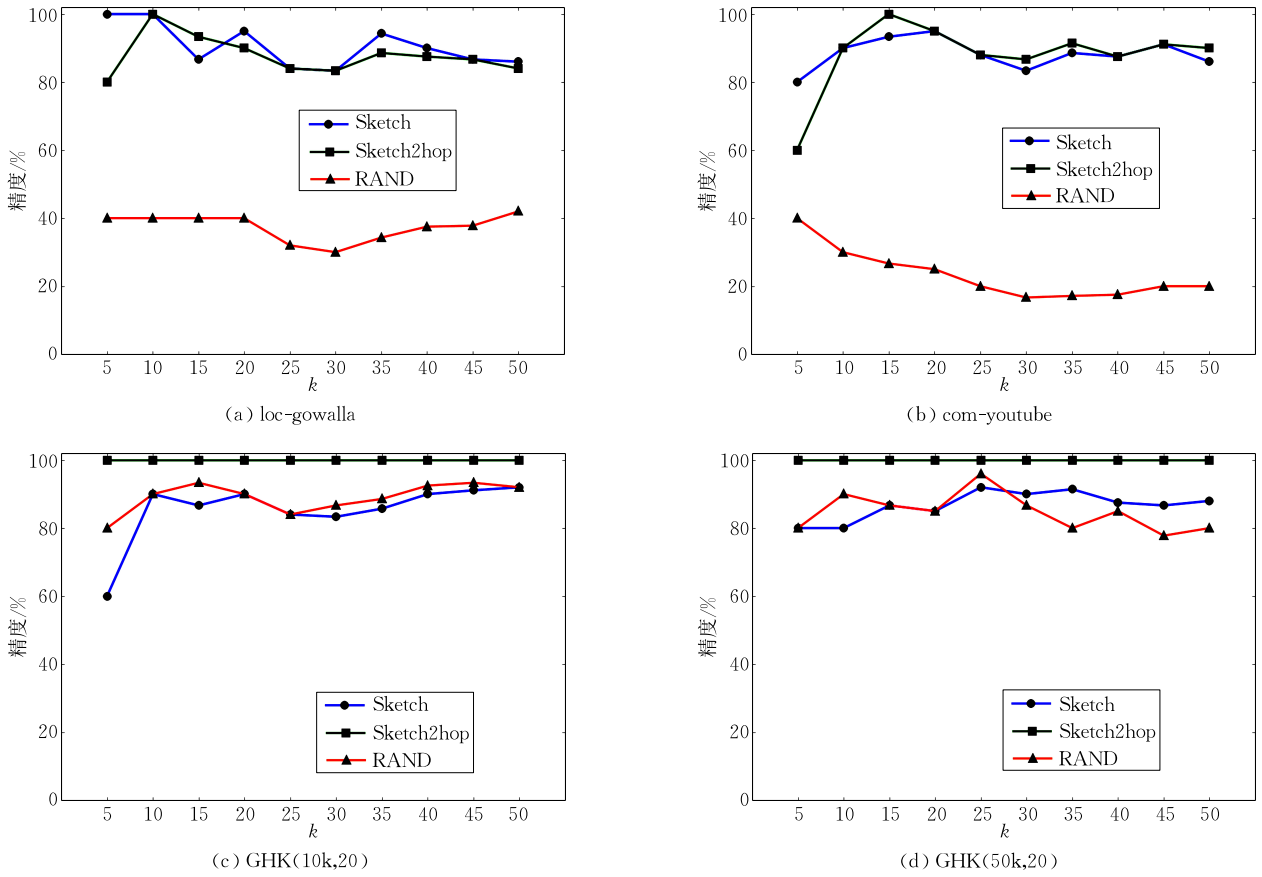


图 4　不同网络数据上的精度与 k 的关系

表 7　Sketch,Sketch2hop 和 RAND 算法在不同数据集上的 MAP@50/%

	GHK(10k,20)	GHK(20k,20)	GHK(30k,20)	GHK(40k,20)	GHK(50k,20)	
Sketch	83.52	84.63	86.08	85.35	86.59	
Sketch2hop	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	
RAND	88.77	89.23	89.30	78.92	83.71	
	ca-HepPh	ca-CondMat	email-Enron	loc-brightkite	loc-gowalla	com-youtube
Sketch	78.52	87.08	86.75	**88.54**	**89.94**	88.47
Sketch2hop	**78.76**	**87.66**	**89.88**	87.65	88.37	**89.53**
RAND	71.61	65.78	14.16	44.34	37.06	22.42

5　相关工作

接下来首先回顾下网络中各类中心性的度量，然后介绍与紧密中心性计算相关的工作.

5.1　多种中心性度量

本节介绍各类中心性的度量方式. 首先，基于度(degree)定义的中心性 C_{deg} 是最流行且最直接的一种定义. 此中心性即为相应节点的度数，一个节点关联的边越多就越重要[9,18].

进一步，基于度的中心性可以解释为统计了节点周围路径长度为 1 的路径数目. 在此基础上，一个直接的泛化就是利用节点周围长度为 K 的路径数目表示节点的重要性，称为 K-Path 中心性. 从路径长度定义类型的不同角度出发，K-Path 中心性也产生了很多变种：基于距离的 K-Path 中心性，边独立的 K-Path 中心性和节点独立的 K-Path 中心性[19].

另外，以通路(walk)的概念为基础，也产生了多种相应的中心性概念. 其中，Katz 中心性[20]用节点出发的通路数目表示，形式化表示为 $\boldsymbol{e}_i^{\mathrm{T}}\left(\sum_{j=1}^{\infty}(\beta A)^j\right)\mathbf{1}$，其中 $\boldsymbol{e}_i$ 是一个第 i 个位置为 1，其他位置为 0 的列向量. β 是一个大于 0 的惩罚系数，为了反映路径越长，其对中心性的贡献越小. 在 Katz 中心性基础上，产生了 Bonacich 中心性，其形式化为 $\boldsymbol{e}_i^{\mathrm{T}}\left(\frac{1}{\beta}\sum_{j=1}^{\infty}(\beta A)^j\right)\mathbf{1}$，并允许 β 取负值. Katz 中心性和 Bonacich 中心性都是

Hubbell 中心性的特殊形式. Hubbell 中心性可以表示为 $\boldsymbol{e}_i^{\mathrm{T}}\left(\sum_{j=1}^{\infty}(\boldsymbol{X})^j\right)\boldsymbol{y}$,其中 $\boldsymbol{X}$ 是矩阵,$\boldsymbol{y}$ 是向量.

上述基于度的中心性定义中,除了度中心性可以在多项式时间复杂度内进行计算以外,其他的中心性均具有较高的时间复杂度,无法进行高效计算.

为了从网络的全局信息衡量一个点的重要性,提出了基于信息流的节点中心性定义. 其中,PageRank[18],HIT[21]等用来衡量一个网页节点在整个网络中的重要性的指标就是经典的基于信息流的中心性定义.

与紧密中心性类似,同样基于最短路定义的一个中心性叫中介中心性(betweenness centrality)[9,22-23],它的具体定义如下:

$$C_b(u)=\sum_{s\neq u\neq t\in V}\frac{\delta_{st}(u)}{\delta_{st}}.$$

其中,δ_{st} 是指节点 s 和 t 之间最短路的数目,而 $\delta_{st}(u)$ 是指节点 s 和 t 之间经过节点 u 的最短路的数目. 由定义可知,中介中心性反映了一个点在信息传播过程中的重要性. 另外,通过上述公式可以发现,中介中心性的计算相对紧密中心性更为复杂,即使计算一个点的中介中心性,都需要先计算全局点对间的最短路. 关于中介中心性计算方法的研究可以参考文献[22-23].

本文重点研究了基于紧密中心性的 top-k 节点搜索问题. 一种标准做法是结合阈值和一定的估算上界值,维护一个前 k 大的节点集合. 当上界值比阈值小的时候,当前的前 k 大节点集合即为最终结果集合. 多数精确算法,利用上述框架进行剪枝,提升算法性能.

5.2 关于紧密中心性搜索的精确算法

接下来,笔者介绍除 Δ-pfs 算法外的关于紧密中心性搜索的精确算法. 一种简单直接的精确算法是利用全局最短路径算法(All-Pairs Shortest Path)计算网络上任意节点对之间的最短路径距离,然后对计算结果进行排序,取排名前 k 的节点即可得到结果. 利用 Floyd-Warshall 算法[24]计算全局点对之间的最短路,算法的时间复杂度为 $O(n^3)$ 和空间复杂度为 $O(n^2)$. 一种优化策略是利用单源最短路 Dijkstra 算法[25]和 Fibonacci 堆[26]计算每一个节点的单源最短路径,从而得到节点的紧密中心性. 优化后的精确算法的时间复杂度是 $O(nm+n^2\log n)$,适合于稀疏的网络. 另外,还有研究工作通过构建索引快速查找一个点对之间的最短路[27];然后,通过实验发现,基于索引的方法比直接利用 Dijkstra 算法计算最短路性能差[27],这是因为紧密中心性的计算需要不断重复访问或计算点对之间的最短路.

然而,目前存在的精确算法,包括 Δ-pfs 算法,在时间和空间复杂度上仍然是多项式的,无法高效处理现实世界中的大规模网络数据. 而本文的工作主要针对新的近似算法的设计和优化.

5.3 关于紧密中心性搜索的近似算法

为了能在大规模网络上高效计算紧密中心性搜索问题,不少相关工作通过设计近似算法对紧密中心性进行估计,然后,基于估计的紧密中心性,获得近似的 top-k 紧密中心性节点.

除了在 2.4 节提到的 RAND 算法之外,Brandes 和 Pich[28]在 RAND 算法基础之上,从如何选择 s 个种子节点的角度深入讨论了此算法并设计了 4 种拓展,分别为 MaxMin,MaxSum,MinSum 和 Mixed.

MaxMin 策略每次选择离之前选中节点越远的点作为下一个种子节点.

MaxSum 策略每次选择与之前选中节点距离之和最大的点作为下一个种子节点.

Mixed 策略,为了避免上述两种策略过度估算距离,利用 round-robin 方式混合循环使用 RAND, MaxMin, MaxSum 策略选择下一个种子节点.

另外,Kang 等人[29]对紧密中心性进行了重新定义,提出了 effective closeness 的概念并利用 MapReduce 并行计算框架进行求解. 其中,effective closeness 是指利用近似的平均距离估算的相应的紧密中心性.

与上述算法不同的是本文提出的近似算法重新考虑了紧密中心性的计算结构,通过以计数的方式进行求解.

5.4 关于紧密中心性搜索的混合算法框架

在处理大规模网络数据时,精确算法无法保证性能,而近似算法虽然高效,但得到的结果却是近似的. 为了解决这个矛盾,Okamoto 等人[30]提出了综合利用精确算法和近似算法的混合计算框架. 在处理 top-k 紧密中心性搜索问题时,此框架首先利用近似算法选出 $k'(k<k'\ll n)$ 个候选节点,然后对每一个候选节点利用精确算法计算相应的准确的紧密中心性,最终选出前 k 个节点作为结果. Okamoto 在 RAND 算法基础上通过设置阈值 δ 选择 k' 个节点,然后把与 k 号节点的估计紧密中心性之差小于 δ 的节点全部纳入候选节点,从而很大概率地保证

最终结果的正确性.

然而，在实际应用中，由于 RAND 算法的精度问题，计算得到候选节点数目过多，甚至能包含整个网络的节点，导致 Okamoto 的混合算法框架性能也不理想. 本文提出的基于 Sketch 的紧密中心性搜索算法可以替代 RAND 算法应用到混合框架内部. 从 RAND 算法和基于 Sketch 的算法实验比较中可以看出，Sketch 算法不仅性能优于 RAND 算法，同时结果精度也高.

6 总　结

社交网络发展的规模越来越大，基于社交网络的分析已经成为各类实际应用不可或缺的分析手段. 本文重点研究了基于紧密中心性的网络上重要节点发现的算法. 为了能够很好地处理大规模网络数据，笔者设计了基于 Sketch 的近似搜索算法，并通过详细的实验验证，基于 Sketch 的近似搜索算法相对于其他精确算法和近似算法在性能和和精度两方面的优势.

在本文工作基础上，笔者将利用并行计算进一步提升算法性能，设计并行的 top-k 紧密中心性搜索框架，另外，考虑到实际中社交网络频繁发生变化，后续也将此算法拓展到处理动态变化的网络.

参 考 文 献

[1] Liu Xin, Li Peng, Liu Jing, Wang Ya-Dan. Centrality for nodes in social networks. Computer Engineering and Applications, 2014, 50(5): 116-120(in Chinese)
(刘欣，李鹏，刘璟，王娅丹. 社交网络节点中心性测度. 计算机工程与应用，2014，50(5)：116-120)

[2] Yuan Wei-Guo, Liu Yun, Cheng Jun-Jun, Xiong Fei. Empirical analysis of microblog centrality and spread influence based on Bi-directional connection. Acta Physica Sinica, 2013, 62(3): 1-10(in Chinese)
(苑卫国，刘云，程军军，熊菲. 微博双向"关注"网络节点中心性及传播影响力的分析. 物理学报，2013，62(3)：1-10)

[3] Wu Xiao-Wei, Xu Fu-Yuan, Song Wen-Guan. Analysis of firm competitors based on centrality degree index of social network. Journal of the China Society for Scientific and Technical Information, 2006, 25(1): 122-128(in Chinese)
(吴晓伟，徐福缘，宋文官. 基于人际网络节点中心度的竞争对手分析. 情报学报，2006，25(1)：122-128)

[4] Krebs V. Mapping networks of terrorist cells. Connections, 2002, 24(3): 43-52

[5] Simsek Ö, Barto A G. Skill characterization based on betweenness//Proceedings of the 23rd Annual Conference on Neural Information Processing Systems. Vancouver, Canada, 2009: 1497-1504

[6] Merrer E L, Trédan G. Centralities: Capturing the fuzzy notion of importance in social graphs//Proceedings of the 2nd ACM EuroSys Workshop on Social Network Systems. Nuremberg, Germany, 2009: 33-38

[7] Cha M, Haddadi H, Benevenuto F, Gummadi K P. Measuring user influence in Twitter: The million follower fallacy//Proceedings of the International Conference on Weblogs and Social Media. Washington, DC, USA, 2010: 1-8

[8] Porta S, Latora V, Wang F, et al. Street centrality and densities of retail and services in Bologna. Italy Environment and Planning B: Planning and Design, 2009, 36(3): 450-465

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

[10] Cui B, Mei H, Ooi B C. Big data: The driver for innovation in databases. National Science Review, 2014, 1(1): 27-30

[11] Shao Ying-Xia, Chen Lei, Cui Bin. Efficient cohesive subgraph detection in parallel//Proceedings of the ACM Special Interest Group on Management of Data. Utah, USA, 2014: 613-624

[12] Shao Ying-Xia, Cui Bin, Chen Lei, et al. Parallel subgraph listing in a large-scale graph//Proceedings of the ACM Special Interest Group on Management of Data. Utah, USA, 2014: 625-636

[13] Olsen P W, Labouseur A G, Hwang Jeong-Hyon. Efficient top-k closeness centrality search//Proceedings of the International Conference on Data Engineering. Chicago, USA, 2014: 196-207

[14] Eppstein D, Wang J. Fast approximation of centrality. Journal of Graph Algorithms and Applications, 2004, 8(1): 39-45

[15] Flajolet P, Martin G N. Probabilistic counting algorithms for data base applications. Journal of Computer and System Sciences, 1985, 31(2): 182-209

[16] David E, Jon K. Networks, crowds, and markets: Reasoning about a highly connected world. Cambridge, UK: Cambridge University Press, 2010

[17] Holme P, Kim B J. Growing scale-free networks with tunable clustering. Physical Review E, 2002, 65(2): 026107-026111

[18] Borgatti S P. Centrality and network flow. Social Networks, 2005, 27(1): 55-71

[19] Borgatti S P, Everett M G. A graph-theoretic perspective on centrality. Social Networks, 2006, 28(4): 466-484

[20] Katz L. A new index derived from sociometric dataanalysis. Psychometrika, 1953, 18(1): 39-43

[21] Kleinberg J M, Authoritative sources in a hyperlinked environment. Journal of the ACM, 1999, 46(5): 604-632

[22] Freeman L C. A set of measures of centrality based on betweenness. Sociometry, 1977, 40(1): 35-41

[23] Brandes U. A faster algorithm for betweenness centrality. Journal of Mathematical Sociology, 2001, 25(2): 163-177

[24] Cormen T, Leiserson C, Rivest R, Stein C. Introduction to Algorithms. 2nd Edition. USA: MGraw-Hill Higher Education, 2001

[25] Dijkstra E W. A note on two problems in connexion with graphs. Numerische Mathematik, 1959, 1(1): 269-271

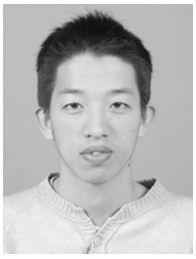
[26] Fredman M L, Tarjan R E. Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM, 1987, 34(3): 596-615

[27] Jin R, Ruan N, Xiang Y, Lee V E. A highway-centric labeling approach for answering distance queries on large sparse graphs//Proceedings of the ACM Special Interest Group on Management of Data. Arizona, USA, 2012: 445-456

[28] Brandes U, Pich C. Centrality estimation in large networks. International Journal of Bifurcation and Chaos, 2007, 17(7): 2303-2318

[29] Kang U, Papadimitriou S, Sun Ji-Meng, Tong Hang-Hang. Centralities in large networks: Algorithms and observations//Proceedings of the SIAM Conference on Data Mining. Arizona, USA, 2011: 119-130

[30] Okamoto K, Chen Wei, Li Xiang-Yang. Ranking of closeness centrality for large-scale social networks//Proceedings of the Frontiers in Algorithmics: Second Annual International Workshop. Changsha, China, 2008: 186-195



SHAO Ying-Xia, born in 1988, Ph. D. candidate. His research interests focus on large-scale graph analysis.

CUI Bin, born in 1975, Ph. D., professor. His research interests include database, social media analysis and large scale graph mining.

MA Lin, born in 1993, undergraduate student. His research interests focus on large-scale graph analysis.

YIN Hong-Zhi, born in 1984, Ph. D. His research interests focus on recommendation system.

Background

The advanced development of various technologies on social network, e-commerce and online education has contributed to an increasing amount of large-scale network data. Among all sorts of network analysis tasks, one basic task is to search important nodes in a network. Closeness centrality is one of popular metrics which measure the importance of a node in a network. To improve the precision of closeness-based top-k search while maintaining the high performance, in this paper, we propose a Sketch-based approximation algorithm for fast searching top-k closeness centrality in a large-scale network. The new algorithm is developed based on a new computation method which calculates the centrality by estimating the number of nodes within a certain distance by a data structure called FM-Sketch. Finally, we compare our Sketch-based algorithm with the state-of-the-art exact and approximation algorithms through extensive experiments. The results demonstrate the advantages of the new solution.

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61272155, 61572039, the National Basic Research Program (973 Program) of China under Grant No. 2014CB340405, and the Shenzhen Gov Research Project under Grant No. JCYJ20151014093505032.