大图上的 SuperSimRank 近似计算方法

张应龙[1,2],夏学文[1,2],余 鹰[2],邓志刚[2]
(1. 闽南师范大学物理与信息工程学院,福建漳州 363000;2. 华东交通大学软件学院,江西南昌 330013)

摘 要: 网络数据具有规模大的特点,而基于关系的相似度计算复杂度高,因此大图上的相似度计算具有很大挑战. 文章针对一个新的相似度度量 SuperSimRank 在大图上的优化计算问题展开研究. 首先提出了阈值过滤技术,使得在计算过程中忽略那些对 SuperSimRank 值影响较小但消耗计算资源的路径值,并通过严格数学证明论证了近似值和准确值的误差;然后在此基础上提出了高效的外存算法,该算法避免了随机访问文件而是通过顺序的读写文件,极大的减少了 I/O 代价;最后实验验证了算法的有效性.

关键词: SuperSimRank; 节点相似度; 大图
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2019)07-1591-05
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2019.07.026

Accuracy Estimate and Optimization Techniques for SuperSimRank Computation on Massive Graphs

ZHANG Ying-long[1,2], XIA Xue-wen[1,2], YU Ying[2], DENG Zhi-gang[2]
(1. *School of Physics and Information Engineering, Minnan Normal University, Zhangzhou, Fujian* 363000, *China*;
2. *School of Software, East China Jiaotong University, Nanchang, Jiangxi* 330013, *China*)

Abstract: Due to the high computational cost and space cost in computing node similarity, it is a challenge when it comes to efficiently computing the similarity on big graphs. In this paper, the following problem will be resolved: how to fast compute SuperSimRank similarity on massive graphs using a single PC. A threshold sieving technology and an external algorithm are introduced. With the help of threshold sieving technology, our external algorithm can efficiently compute the similarity on massive graphs. Experimental results demonstrate the efficiency of the computation.

Key words: SuperSimRank; nodes similarity; big graph

1 引言

节点相似度计算作为图数据挖掘中的基本问题,已经得到广泛的应用[1~3],其中,Personalized PageRank (PPR)[4]和 SimRank(SR)[5]为代表性度量,工作[6]巧妙融合了 PPR 和 SR 框架,设计了一个新度量 SuperSimRank(SSR),该度量更加全面的基于网络结构特性来进行定义节点相似度.

因为节点相似度计算复杂度高,其在大图上的计算具有很大的挑战性. 分布式系统是处理大数据的理想选择,但图算法在分布式系统中存在节点间通信代价高,环境的搭建,编写分布式程序比较复杂,难于维护、优化,成本高等问题[7,8]. 对此,研究者近年来提出在单台个人计算机上设计基于磁盘的系统[7,9~11]处理大规模网络数据,而这些系统缺乏针对基于关系的相似度进行优化,所以直接利用这些系统来计算 SSR 有困难,文献[12]认为单机上数据分析技术[7]代表大数据技术发展的一个前沿方向. 因此,有必要研究 SSR 在大规模数据上的单机处理算法. 本文主要贡献包括:(1)提出阈值过滤技术,并且通过数学证明了该阈值过滤技术;(2)提出基于外存的计算 SSR 框架算法,通过变换 SSR 公式,使得每次计算采用顺序读取所需数据,从而最大限度减少 I/O 次数.

2 SuperSimRank 简介

给定有向图 $G=<V,E>$,V 表示对象的集合,E 表示对象之间的关系;$I(v)=\{u|$有向边$<u,v>\in E\}$ 表示图中节点 v 的入度邻居集合,其中的元素表示为 $I_i(v)(1\leqslant i\leqslant|I(v)|)$;$O(v)=\{u|<v,u>\in E\}$ 表示图中节点 v 的出度邻居集合,其中的元素表示为 $O_i(v)(1$

收稿日期:2018-01-22;修回日期:2019-01-15;责任编辑:孙瑶
基金项目:国家自然科学基金(No. 61762036, No. 61663009, No. 61563016);江西省自然科学基金(No. 20171BAB202012, No. 20181BAB202023);江西省交通厅科研项目(No. 2017D0038);江西省教育厅科技项目(No. GJJ180322)

$\leqslant i \leqslant |O(v)|$).

节点 a 和 b 的 SSR 公式(详情见文献[6])为:

$$M_{k+1}(a,b) = \frac{c}{|I(a)||I(b)|}\sum_{i=1}^{|I(a)|}\sum_{j=1}^{|I(b)|}M_k(I_i(a),I_j(b)) + T_{k+1}(a,b) \tag{1}$$

这里

$$T_{k+1}(a,b) = (1-c)\left(\sum_{\substack{\tau:a\to b\\ l(\tau)\leqslant k+1}} c^{l(\tau)}p(\tau) + \sum_{\substack{\tau:b\to a\\ l(\tau)\leqslant k+1}} c^{l(\tau)}p(\tau)\right)\Big/2$$

上述公式中 c 为常数取值 0.5,且 $a\neq b$,$\tau:a\to b$ 为 a 到 b 的单向路径,$l(\tau)$ 为该路径的长度,$p(\tau)$ 是为 a 到 b 的单向路径的概率. 节点对节点本身的相似度为 1. 初始值 $M_0(a,b)$ 当 $a=b$ 为 1,否则为 0.

文献[6]同时给出了与 SSR 公式(见式(1))等价的另一种形式:

$$M_{k+1}(a,b) = M_k(a,b) + \sum_{\substack{(d+l(\tau'))=k+1\\ \tau':(x,y)\to(a,b)}} c^{l(\tau')}p(\tau') + \frac{1-c}{2}\sum_{\substack{l(\tau)=k+1\\ \tau:a\to b\\ \cup\tau:b\to a}} c^{l(\tau)}p(\tau) \tag{2}$$

其中,d 表示 x 到 y 或 y 到 x 的单向路径的长度,当 $x=y$ 时,d 为零;$(x,y)\to(a,b)$ 表示两个人分别从 a 和 b 逆向同步游走且同时分别抵达点 x 和 y 对应的路径.

假设 e,f 分别是 a,b 入度邻居节点,令

$$u_k(e,f) = \sum_{\substack{l(\tau)=k\\ \tau:e\to f}} c^{l(\tau)}p(\tau)$$

$$w_k(e,f) = \sum_{\substack{l(\tau')=k\\ \tau':(x,y)\to(e,f)}} c^{l(\tau')}p(\tau') + \frac{1-c}{2}(u_k(e,f)+u_k(f,e))$$

$$s_{k+1}(a,b) = \frac{c}{|I(a)||I(b)|}\sum_{\substack{e\in I(a)\\ f\in I(b)}} w_k(e,f)$$

这时式(2)可以写为:

$$M_{k+1}(a,b) = M_k(a,b) + s_{k+1}(a,b) + \frac{1-c}{2}(u_{k+1}(a,b)+u_{k+1}(b,a)) \tag{3}$$

由上可知,每次迭代,新增加的值为

$$w_{k+1}(a,b) = s_{k+1}(a,b) + \frac{1-c}{2}(u_{k+1}(a,b)+u_{k+1}(b,a)) \tag{4}$$

基于式(3)(4),将提出阈值过滤技术和外存算法.

3 阈值过滤技术

针对式(3),在第 $k+1$ 次迭代时,过滤那些对 SSR 贡献小且长度为 $k+1$ 的路径,从而减少其在后续计算中带来大的计算资源浪费. 换言之,当某些节点对的 $s_{k+1}(*,*)$ 与 $u_{k+1}(*,*)$ 的值小于指定阈值时,令其取 0,不再参与后续的计算,而最终舍掉的值累积起来不超过 $\Delta(K)$.

令 $s'(*,*)$,$u'(*,*)$,$w'(*,*)$ 和 $M'(*,*)$ 为 $s(*,*)$,$u(*,*)$,$w(*,*)$ 和 $M(*,*)$ 对应的估计值. 现讨论如何估算新增的路径贡献值:$s(*,*)$ 和 $u(*,*)$.

$s(*,*)$ 的近似估算方案如下:对 $\delta\geqslant 0$ 和,当 $k>0$ 时若 s 的值大于 $2c^k\delta$(或 $k=0$ 时,右端大于 δ),有:

$$s'_{k+1}(a,b) = \frac{c}{|I(a)||I(b)|}\sum_{\substack{e\in I(a)\\ f\in I(b)}} w'_k(e,f) \tag{5}$$

否则

$$s'_{k+1}(a,b) = 0 \tag{6}$$

$u(*,*)$ 的近似估算方案:对 $\mu_k = k\frac{c^{k-1}}{1-c}\delta$,当 $k>0$ 时,若 $u(*,*)$ 的值大于 $\frac{\mu_{k+1}}{k+1}$,有

$$u'_{k+1}(a,b) = c\sum_{f\in I(b)}\frac{u'_k(a,f)}{|O(f)|} \tag{7}$$

否则

$$u'_{k+1}(a,b) = 0 \tag{8}$$

对于以上估算方案,通过以下 3 个引理来阐述对应的估计值的误差.

引理 1 $u_k(a,b)$ 与 $u'_k(a,b)$ 之间的误差为 $u_k(a,b)-u'_k(a,b)\leqslant\mu_k(k\geqslant 1)$.

引理 2 $s_k(a,b)$ 与 $s'_k(a,b)$ 之间的误差为 $s_k(a,b)-s'_k(a,b)\leqslant\eta_k$,这里 $\eta_1=\delta$,$\eta_k=\left(\sum_{i=1}^{k+1} i-2\right)c^{k-1}\delta(k\geqslant 2)$.

由于篇幅所限,略去引理 1、2 证明,如需请与作者联系.

引理 3 当 $k\geqslant 1$ 时,$M_k(a,b)$ 与 $M'_k(a,b)$ 之间的误差为 $M_{k+1}(a,b)-M'_{k+1}(a,b)\leqslant\Delta(k)$,这里 $\Delta(k)=\sum_{i=1}^{k}(\eta_i+(1-c)\mu_i)$.

证明 因为

$$M_k(a,b) = \sum_{i=1}^{k} w_i(a,b) = \sum_{i=1}^{k}\left(s_i(a,b)+\frac{1-c}{2}(u_i(a,b)+u_i(b,a))\right)$$

根据引理 1,2 可得:

$$M_k(a,b)-M'_k(a,b) = \sum_{i=1}^{k}\left(\begin{array}{l} s_i(a,b)-s'_i(a,b)\\ +\frac{1-c}{2}\left(\begin{array}{l}u_i(a,b)-u'_i(a,b)\\ +u_i(b,a)-u'_i(b,a)\end{array}\right)\end{array}\right) \leqslant \sum_{i=1}^{k}(\eta_i+(1-c)\mu_i$$

证毕.

令 K 为最大的迭代次数,指定阈值 $\Delta(K)$,由引理 3 知

$$\Delta(K) = \sum_{k=1}^{K}(\eta_i + (1-c)\mu_i) = \left(2 + \sum_{k=1}^{K}\left(\frac{k(k+5)}{2} - 1\right)\right)\delta$$

因此可以确定 δ 的值 $\delta = \dfrac{\Delta(K)}{2 + \sum_{k=1}^{K}\left(\frac{k(k+5)}{2} - 1\right)}$.

$\Delta(K)$ 是 $M_k(a,b)$ 与 $M'_k(a,b)$ 之间的误差上界,真实误差值小于 $\Delta(K)$,例如,当 $K=5$ 时,真实误差与 $\Delta(K)$ 相差一个数量级. 特别的,当 $\Delta(K)=0$,$M_k(a,b)=M'_k(a,b)$.

4　基于外存的算法

整个思想是图数据和中间结果储存在外存,每次从磁盘文件顺序读入数据块,然后利用有限的主存处理数据,处理后又顺序的把数据储存在磁盘文件中,直到得出最终的结果. 具体计算时也是基于式(3)(4)展开.

磁盘文件 edge 储存图数据,每行的格式为 $(a,b_1,p_1,\cdots,b_t,p_t)$,其中 $<a,b_i>(1\leqslant i\leqslant t)$ 为有向边 a 指向 b_i,对应的 p_i 为 $1/|I(b_i)|$. 其余的磁盘文件用来储存中间和最终结果,它们每行的格式都为 $(a,b_1,v_1,\cdots,b_t,v_t)(1\leqslant i\leqslant t)$,这里 v_i 为节点对 (a,b_i) 对应的值. 磁盘中的所有内容按 a 从小到大的顺序排序.

算法 1 是基于外存的算法,它包含了三个子函数,每个子函数产生若干个磁盘文件用来储存每次迭代的中间结果.

算法 1　External algorithm

```
Input: edge, c, Δ, M // M is maximum iteration
Output: SSR score // SuperSimRank Result
1: read data from dege and achieve M'_1, w'_1 and u'_1 then save these values into files Mf, wf and uf;
2: for k = 2 to M do
3:    uf, muf ← obtainUniValues(edge, uf, u_{i,} c);
4:    pwf ←obtainPartialValues(edge, wf, c);
5:    Mf, wf ←obtainValues(edge, Mf, muf, pwf, δ);
```

算法 2 描述了子函数 obtainUniValues,其所产生的磁盘文件 *uf* 所包含的值由如下公式所得:$u_{k+1}(a,b) = c\sum_{f\in I(b)}\frac{u_k(a,f)}{|O(f)|}$(行 1 ~ 13),即每行的 v_i 值为 $u_{k+1}(b_i,a)$,其中行 13 采用式(7)(8)进行阈值过滤. 该子函数所产生的另一个磁盘文件 *muf* 所包含的 v_i 值为 $v_i = u_{k+1}(b_i,a) + u_{k+1}(a,b_i)$ 且 $a<b_i$(行 14 ~ 21).

算法 2　obtainUniValues

```
Input: edge, uf, c, μ_i // μ_i: proposition 1
Output: uf, muf
1: e←edge. getNextLine( );
2: u←uf. getNextLine( );
3: while ! edge. eof( ) && ! uf. eof( ) do
4:    if e. a = = u. a then
5:       for i←1 to T_1 do
6:          for j←1 to T_2 do
7:             U[e. b_i][e. b_j] ←U[e. b_i][e. b_j] + c * u. v_i/|O(e. a)|;
8:             if buffer U is full then
9:                sort the U by the e. b_i and write it to a temp file
10:   e←edge. getNextLine( ); u←uf. getNextLine( );
11:      else if e. a < u. a then e←edge. getNextLine( );
12:      else    u←uf. getNextLine( );
13: sorted temp files and filter small values based on e. q. (7)(8) to generate a new file uf;
14: while ! uf. eof( ) do
15:   u←uf. getNextLine( );
16:   for i←1 to T_1 do
17:      if u. a < u. b_i then MU[u. a][e. b_i] ←MU[u. a][e. b_i] + u. v_i
18:   else  MU[e. b_i][u. a] ←MU[e. b_i][u. a] + u. v_i
19:   if buffer U is full then
20:            sort the MU by first node and write it to a temp file
21: merged sorted temp files to generate a new file muf;
```

由于 $s_{k+1}(a,b) = \dfrac{c}{|I(a)||I(b)|}\sum_{\substack{e\in I(a)\\ f\in I(b)}} w_k(e,f)$ 可改写为如下两个公式:

$$pw_{k+1}(I_j(b),a) = \frac{c}{|I(a)|}\sum_i w_k(I_i(a),I_j(b)),$$

$$s_{k+1}(a,b) = \frac{1}{|I(b)|}\sum_j pw_{k+1}(I_j(b),a).$$

因此可以分两步获取 $s_{k+1}(a,b)$. 第一步,利用子函数 obtainPartialValues(算法 3)获取 $pw_{k+1}(a,I_j(b))$,并把该值保存在文件 *pwf*,因此文件中每行的 v_i 值为 $pw_{k+1}(a,b_j)$.

算法 3　obtainPartialValues

```
Input:  edge, wf, c
Output: pwf
1: e←edge. getNextLine( );
2: u←uf. getNextLine( );
3: while  ! edge. eof( ) && ! wf. eof( ) do
4:    if e. a = = u. a then
5:       for i←1 to T_1 do
6:          for j←1 to T_2 do
7:             U[u. b_j][e. b_i]←U[u. b_j][e. b_i] + c * u. v_j *  e. v_i;
8:             if buffer U is full then
9:                sort the U by the e. b_j and write it to a temp file
10:      e←edge. getNextLine( );  u←uf. getNextLine( );
11:      else if e. a < u. a then e←edge. getNextLine( );
12:      else     u←uf. getNextLine( );
13: merged sorted temp files to generate a new file pwf;
```

算法4读取以下三个文件内容 *tempf*,*muf* 和 *Mf*,然后基于式(5)(6)(4)(3)计算 $s'_{k+1}(a,b)$,$w'_{k+1}(a,b)$ 和 $M'_{k+1}(a,b)$. 新产生的磁盘文件 *wf* 和 *Mf* 中对应的 v_i 值分别为 $v_i(a,b_i)=w'_{k+1}(a,b_i)$ 与 $v_i(a,b_i)=M'_{k+1}(a,b_i)$.

算法4 obtainValues

```
Input:edge,pwf,muf,Mf,δ
Output:wf,Mf
//obtain s_{k+1} and filter small values based on e. q. (5)(6)
1:tempf←LA+pwf
  //obtain w'_{k+1} and M'_{k+1}:
2:s←tempf. getNextLine(); u←uf. getNextLine();
  m←Mf. getNextLine()
3:while ! edge. eof() || ! muf. eof()|| ! Mf. eof() do
4:  if s. a = =u. a && u. a! =LONG_MAX then
5:    temp←s+u;s←tempf. getNextLine();
6:    u←uf. getNextLine();
7:  else if s. a<u. a then temp←s; s←tempf. getNextLine();
8:  else temp←u; u←muf. getNextLine();
9:  write temp into file wf;
10:  if tempf. eof() then s. a←LONG_MAX;
11:  if wf. eof() then u. a←LONG_MAX;
12:  if m. a = =temp. a then
13:      write m+temp into file Mf1;m←Mf. getNextLine();
14:  else if m. a<temp. a then
15:      write m into file Mf1; m←Mf. getNextLine();
16:  else write m into file Mf1;
17:  if Mf. eof() then m. a←LONG_MAX;
18:delete Mf,and Mf1 is renamed Mf;
```

5 实验

实验环境:所有实验在 CPU 为 i3-550,主存为 4G 的 PC 上进行,操作系统为 windows 7,C++实现代码.

实验主要考察基于外存的算法处理大图的效率. 选取了3个真实的大图(http://snap.stanford.edu/data/index.html),相关统计信息见表1.

表1 三个大图的统计信息

	Web-google	web-Standford	Amazon0505
Nodes	875,713	281,903	410,236
Edges	5,105,039	2,312,497	3,356,824

表2是在这三个大图上准确计算 SSR 值的运行时间(文中未采用阈值过滤技术的外存算法),从表中可知,准确计算 SSR 值是非常耗时.

表2 4次迭代后准确计算 SSR 值的运行时间

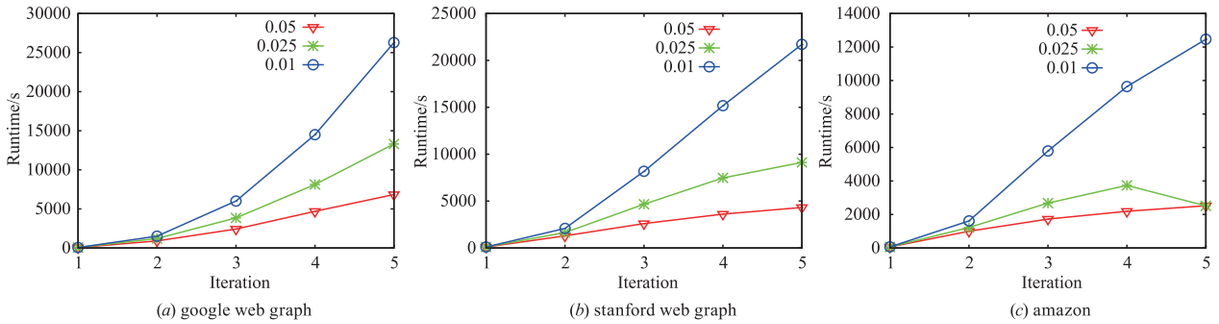
	Web-google	web-Standford	Amazon0505
Runtime/s	274,043	146,515	191,796

对于采用阈值过滤技术的外存算法,设最大的迭代次数 $K=5$,$\Delta(K)$ 分别设为 0.05,0.025 以及 0.01.

图1是算法在三个大图上 $\Delta(K)$ 取不同值的运行时间,对于主存为4G的PC而言,该算法处理大图的运行速度是非常快. 以 $\Delta(K)=0.01$ 为例,比较其在第四次迭代后的运行时间与表3中准确计算 SSR 值的时间,在 google web graph 上,估算算法比准确算法运行时间提高了18倍之多,对于 stanford web graph,运行时间提高了9倍多,对于 amazon 图运行时间提高了19倍多,所以阈值过滤技术在提高处理数据的运行时间方面起着重要的作用,另一方面,$\Delta(K)=0.01$ 时第四次迭代后的估算值与真实值之间的平均真实误差不超过0.001(见表3),因此估算算法无论运行速度还是误差要求均可达到实际处理数据的需求了.

表3 4次迭代后的平均真实误差

	Web-google	web-Standford	Amazon0505
Difference	0.0085	0.0079	0.009



(*a*) google web graph (*b*) stanford web graph (*c*) amazon

图1 算法在三个大图上Δ(*K*)取不同值的运行时间

6 相关工作

使用 SR 与 PPR 计算大规模的相似度的时间复杂性高、I/O 开销大,因此有大量的研究提高 SR 和 PPR 计算性能[13~16],然而 SSR 与 SR、PPR 公式有本质区别[6],这样的区别导致无法用现有的这些技术来优化 SSR 计算性能.

网络的表征学习(Embedding)[17]是与节点相似度有联系但又不同的非常重要的一个研究方向. 其本质上学

习网络特征来构造节点的向量表示,SSR 通过更加全面考虑网络特征提出新的节点相似度度量,这种思想我们相信可以借鉴到表征学习中,而且本文阈值过滤技术可以加速学习过程. 因此,表征学习如何借鉴优秀的节点相似度思想来提高准确性是值得研究的一个方向.

7　总结

本文针对 SSR 优化计算问题展开,首先对 SSR 进行了简介,并对相似度计算公式进行了等价变形,然后在变形后的公式基础上提出了阈值过滤技术,接着提出了基于过滤技术的外存算法,最后通过实验验证了该算法的有效性.

参考文献

[1] ASHISH G, et al. The who-to-follow system at twitter: Strategy, algorithms, and revenue impact [J]. Interfaces, 2015,45(1):98 – 107.

[2] FORTUNATO S. Community detection in graphs [J]. Physics Reports,2010,486(3 – 5):75 – 174.

[3] 俞春花,等. 基于上下文相似度和社会网络的移动服务推荐方法[J]. 电子学报,2017,45(6):1530 – 1536.
YU Chu-hua, et al. Mobie service recommendation based on context similairity and social network[J]. Acta Electronica Sinica,2017,45(6):1530 – 1536. (in Chinese)

[4] JEH G, WIDOM J. Scaling Personalized Web Search [OL]. http://www2003. org/cdrom/papers/refereed/p185/html/p185-jeh. html. 2018 – 12 – 01.

[5] JEH G, WIDOM J. SimRank: A Measure of Structural-Context Similarity [OL]. http://ilpubs. stanford. edu:8090/508/1/2001-41. pdf. 2018 – 12 – 01.

[6] 张应龙,等. 信息网络中一个有效的基于链接的结点相似度度量[J]. 软件学报,2014,25(11):2602 – 2615.
ZHANG Y, et al. Effective link-based measure of node similarity on information networks [J]. Journal of Software,2014,25(11):2602 – 2615. (in Chinese)

[7] KYROLA A, et al. Graphchi: Large-scale graph computation on just a PC[A]. Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation[C]. US:USENIX Association,2012. 31 – 46.

[8] 吴城文,等. 从系统角度审视大图计算 [J]. 大数据,2015,1(3):39 – 47.
WU C. et al. Reviewing large graph computing from a system perspective[J]. Big Data Research,2015,1(3):39 – 47. (in Chinese)

[9] CHENG J, et al. VENUS: Vertex-centric streamlined graph computation on a single PC[A]. Proceedings of IEEE 31st International Conference on Data Engineering [C]. New York:IEEE,2015. 1131 – 1142.

[10] ZHU X, et al. GridGraph: Large-scale graph processing on a single machine using 2-level hierarchical partitioning [A]. Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation[C]. US: USENIX Association,2015. 375 – 386.

[11] CHI Y, et al. NXgraph: An efficient graph processing system on a single machine[A]. Proceedings of IEEE 32nd International Conference on Data Engineering[C]. New York:IEEE,2016. 409 – 420.

[12] 吴甘沙. 大数据技术发展的十个前沿方向 (中)[J]. 大数据,2015,1(2):109 – 117.
WU G-S. Ten fronties for big data technologies(Part B)[J]. Big Data Research,2015,1(2):109 – 117. (in Chinese)

[13] LIZORKIN D, et al. Accuracy estimate and optimization techniques for simrank computation[J]. VLDB Journal, 2010,19(1):45 – 66.

[14] YU W, et al. Dynamical SimRank search on time-varying networks[J]. VLDB Journal,2018,27(1):79 – 104.

[15] WANG S, et al. FORA: Simple and effective approximate single-source personalized pagerank[A]. Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery in Data Mining[C]. New York: ACM,2017. 505 – 514.

[16] MINHAO J, et al. READS: a random walk approach for efficient and accurate dynamic SimRank[J]. Proceedings of VLDB Endow,2017,10(9):937 – 948.

[17] PEROZZI B, et al. Deepwalk: online learning of social representations[A]. Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining[C]. New York:ACM,2014. 701 – 710.

作者简介

张应龙　男,1979 年 1 月生. 博士,现为闽南师范大学副教授. 从事网络分析与挖掘的有关研究.

E-mail:zhang_yinglong@126com