计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190800122

历史图上基于 CSR 结构的 PageRank 算法

潘培贤　邹兆年　李发明
哈尔滨工业大学计算机科学与技术学院　哈尔滨 150001
(pan_rose@yeah.net)

摘　要　近年来,学者们对静态图的研究越来越全面、深入,已经形成了完善的理论体系。但是,对于生活中的一些应用问题,如社交网络中不断变化的关系等,使用静态图表示此类动态变化的关系似乎显得有些乏力。而历史图可以表示动态的变化。PageRank 算法是用于衡量网页重要程度的算法,而网络中不断有网站新建或删除,这样的网络用历史图来表示更为合适,因此考虑在历史图上利用 CSR(Compressed Sparse Row)结构实现 PageRank,使得程序能够给出几个目标时间上各网站的评分,进而能够提供网站评分的变化情况,给出网站影响力趋势的预测。在 Wekipedia 提供的网页互相连接的 Hyperlink networks 数据集上,将所提方法与在链表上实现 PageRank 算法做比较,结果显示其性能大大优于使用链表的结构,并且随着数据规模和目标时间规模的增大,其优势将会越来越明显。

关键词:PageRank;CSR 结构;历史图
中图法分类号　TP311

CSR-based PageRank Algorithm on Historical Graphs

PAN Pei-xian,ZOU Zhao-nian and LI Fa-ming
School of Computer Science and Technology,Harbin Institute of Technology,Harbin 150001,China

Abstract　In recent years,the research on static graphs has become more and more comprehensive and in-depth,and a perfect theoretical system has been formed. However,for some application issues in life,such as the changing relationships in social networks,it seems a bit weak to use static graphs to show that such moments are changing. Historical graph can be used to represent dynamic changes. The PageRank algorithm is an algorithm for measuring the importance of a web page,and there are constantly new or deleted websites in the network. Considering all these factors,such a network is quite appropriate to be represented by historical graphs. Therefore,this paper considers using the CSR (Compressed Sparse Row) structure to implement PageRank on the history graph,so that the program can give the scores of each website at several target times. In turn,it can provide changes in website ratings and give forecasts of website influence trends. Comparing its performance with the PageRank algorithm implemented on the linked list based on the hyperlinks network dataset provided by Wekipedia,the results show that its performance is much better than the use of linked list structure,and its advantages will become more and more obvious as the data size and target time scale increase.

Keywords　PageRank,Compressed Sparse Row structure,Historical graph

1　引言

1.1　课题背景及研究的目的和意义

大数据是信息化发展到一定阶段的产物。大数据不仅仅是数量上的大,在其背后还隐藏着各个行业、领域错综复杂的关系,而这些关系往往蕴含着商业价值与科研价值[1]。“图”是一种很好的表示关系的数据结构,它能够有效且直观地描述现实生活中各个事物之间的复杂关系以及各个事物本身所具有的一些性质。在现实生活中存在的对象可被抽象为图中的顶点,对象和对象之间的某种或者多种关系可被抽象为顶点与顶点之间的单边或者多边,如社交网络中用户与用户之间的好友关系等。

传统的图数据是静态的,只能表示一个单一时刻的数据。但是对于真实世界中每时每刻都在变化的生命体及其组成来说,静态图注定无法表示。Przytycka 等认为在未来的工作中,从静态网络分析提升到动态网络分析是必然的[2]。同理,

到稿日期:2019-08-26　返修日期:2019-10-18　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金面上项目(61672189);国家自然科学基金重点项目(61532015)
This work was supported by the Surface Project of National Natural Science Foundation of China (61672189) and Key Program of National Natural Science Foundation of China (61532015).
通信作者:邹兆年(znzou@hit.edu.cn)

对于每时每刻都有页面不断加入和退出的互联网，我们需要一个更好的结构对其进行表示。

1.2 国内研究现状

传统的 PageRank 推荐算法需要全图迭代到收敛，因此其时间复杂度非常高，严重影响了推荐的效率。Cao 深入分析了 PageRank 的关键技术，并对其存在的商业问题进行了思考[3]。为了减少算法的耗时，Chang 等[4]在 PageRank 算法的迭代过程中加入可控制迭代次数的参数 b 和一个用于修剪结果向量的阈值 α；然后针对主题相关性问题，使用了归一化的邻接矩阵的特征值与特征向量来评估结点之间的距离，从而产生最终的推荐列表，列表中对象主题的相关性较高。对于 APP 搜索来说，虽然按照关键词搜索出来的应用主题相关性比较高，但其质量参差不齐。因此，Li 等[5]在 PageRank 算法的基础上引入保持时间因子，使用户保存时间较短的 APP 逐渐悬沉，而保存时间长的 APP 能快速飘浮。Pedroche 等[6]认为 PageRank 是马尔可夫链(1 阶)的静止状态，它利用先前状态的知识来转换系统的状态。他们利用“个性化向量”来让 PageRank 可以偏向某个结点，而且将 PageRank 和多路复用网络相结合，定义了 Multiplex PageRank。

对于传统的静态图完善的理论体系结构来说，历史图仍在发展中。近几年，越来越多的学者开始关注历史图相关方面的研究。

在文献[7]中，历史图被描述为一系列的静态图序列。由于面向大规模动态图的可达查询研究较少，且尚存在压缩困难以及图结构待优化等问题，Ding 等[8]提出了一种支持大规模数据的基于改进哈夫曼编码的可达查询处理方法。首先，对预处理图进行结构上的两次压缩，得到双压缩图；其次，基于双压缩图提出一种前缀 label 索引，该索引能够有效表达结点间的可达关系；最后，提出双压缩图的演进和可达查询处理及优化算法。面向结构变化的动态图匹配问题最早由 Wang 等于 2009 年提出，他们构建邻结点树(Nearest-Neighbors Tree，NNT)[9]并依此对匹配候选集进行过滤，从而有效减少了假阳性匹配结果的产生。这之后的代表性算法有 IncIso-Match[10]和 SJ-Tree[11]等，它们对进一步提升子图匹配的执行效率提供了不同策略。文献[12]指出：给定一个网络图，计算给定两点之间的距离是一个重要的问题。其对最先进的基于空间相干和基于顶点重要性的方法进行了综合比较；同时使用具有多达 2000 万个顶点的各种真实道路网络，分别计算了两种技术的预处理时间、空间消耗和查询效率，以此来评估两种技术。

2 PageRank 算法

根据文献[13]的思想，若有一个网页 u，那么定义 F_u 为网页 u 指向的网页集合，B_u 为指向网页 u 的网页集合。$N_u=|F_u|$，即网页 u 的出链数目。c 是标准化因子(使得所有网页权重之和为常数)。

首先，定义一个简化版的 PageRank 公式：

$$PR(u)=c\sum_{v\in B_u}\frac{PR(v)}{Nv} \tag{1}$$

这个简化的排名函数存在一个小问题：对于只有链入而无链出的结点，在迭代时，该循环会不断积累权重，而从不发出权重。对于这类没有出度的网页，设置其对所有网页都有出度。

PageRank 迭代计算的过程如算法 1 所示。

算法 1 PageRank 算法

输入：Graph G

输出：PageRank Value

```
1. Initialize:float array PR[]
2. for i=1 up to max_interation do
3.    change=0
4.    for node∈G.nodes() do
5.       PR(node)=Σ_{v∈G.nodes} PR(v)/Nv
6.       PR(node) +=(1-c)/N
7.       change +=|PR_old-PR_new|   /* 累计该轮与上轮差值 */
8.    endfor
9.       if change < threshold then
10.         break
11.      end if
12. end for
```

然而，PageRank 忽视了主题性和相关性。例如，“苹果”对于不同主题蕴含着不同的意思。为弥补这些问题，个性化的 PageRank[14]与主题敏感的 PageRank[15]应运而生。如果将 PageRank 和 IF-TDF 结合起来[16]，则能够基于文档内容与用户查询的相似性对文档进行排名。同时，PageRank 也可以应用到电力系统[17]和军事通信系统中[18]。

3 历史图

历史图数据，又可以称为临时图数据(Temporal Graph)和动态图数据(Dynamic Graph)。其顶点可表示某个实体对象，例如网页、通信双方等；边可表示对象的关系。图的边或者点上存在时间戳，表示边或者点出现或消失的时间。

历史图结构会随时间的变化而变化，其动态性主要表现为点和边的不确定性。

历史图数据由诸多个 Δ 构成，每个 Δ 可以看作一个四元组(v_1，v_2，$operation$，$timestamp$)，分别表示结点 1、结点 2、操作类型(增\减)和时间戳，以下算法用到的四元组数据就是此结构。

在对历史图进行相关计算时，为了避免目标时间距离开始时间过长，我们在线下保存一个时间线中部的快照(见图 1)，用以表示该时刻的图结构。如果查询时间大于 snapshot 的时间，那么直接读取 snapshot 的内容即可获得整个时间线半程时的状态。

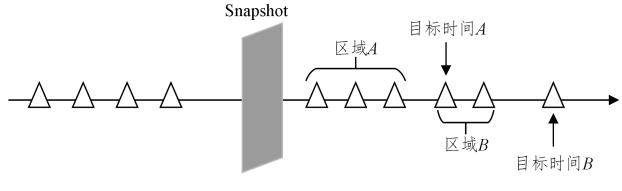


图 1 历史图结构

Fig. 1 Component of historical graph

4 CSR 结构

4.1 基本结构

CSR 结构是一种用两个数组表示图数据的数据结构。点数组记录每个点的邻居在另一个数组(边数组)中的位置;边数组用于存放邻居的编号。如此,可以根据两个数组中的内容还原图中的所有数据。

4.2 结构特点与优势

在空间方面,链表的每个元素包含两个部分,一个部分是存储的数据,另一个部分是指向下一个结点地址的指针,因此在表示图时需要多占用 m 个指针所需要的空间(m 表示边的数目)。而使用 CSR 结构时,只需要 $m+n$ 个 int 型的空间即可表示同样的图(n 表示点的数目)。因此,CSR 结构更节省空间。

在时间方面,由于数组的元素在内存空间中是连续存放的,而链表存放的空间可以是连续的,也可以是不连续的,因此在遍历数据获取图信息时,CSR 结构会比链表快。同时,在处理边的删除操作时,链表需要进行查找、删除、重新指向的操作,比较费时。

5 使用 CSR 结构实现 PageRank

5.1 LinkedList Method

LinkedList Method 使用最直观的链表维护一个图,对每个 delta 操作:如果是加边,则到对应的点指向的链表处加上新邻居结点;如果是减边,则在该点的邻居链表中找到对应邻居,删除结点后连接两端。在遇到需要计算 PageRank 的时间点时,根据当前的图,可以获取所需要的所有信息(结点数目、终止点数目、结点邻居等),然后调用 PageRank 函数进行计算。

在算法 2 中,首先获取历史图的四元组 v(第 2—3 行);对增/减操作进行不同的处理(第 7—11 行);如果该时间戳大于目标时间(即链表已是该目标时间的状态),则将目标时间的所有操作完成后计算 PageRank 值(第3—6 行)。

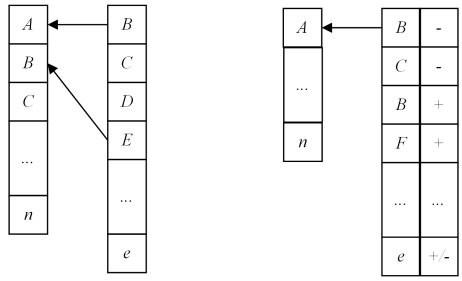
算法 2 链表方法

```
输入:Historical graph data v[ ],target time array t[]
输出:PageRank Value in every target time
1. while True do
2.    reload v / * 读取四元组数据 * /
3.    node1,node2,ope,time←v[0],v[1],v[2],v[3]
4.    if time is one of target times then
5.       calPR() / * 计算 PageRank 值 * /
6.    end if
7.    if ope=1 then
8.       addNode(node1,node2)/ * 添边 * /
9.    else
10.      deleteNode(node1,node2) / * 减边 * /
11.   end if
12. end while
```

在算法 2 中,首先初始化 *PageRank* 值(第 1 行),其中对终止结点(未指向任何边的结点)赋值 $\frac{end\,point\_count}{node\_count^2}$,而对非终止结点赋值 $\frac{1}{node\_count}+\frac{end\,point\_count}{node\_count^2}$。这可以理解为一开始将总值 1 分成 *node_count* 份,而终止结点由于没有出度,跳到该点后会随机跳转到任意结点,所以将其 *PageRank* 值分给所有结点(包括其他终止结点)。之后进行每一轮的迭代,收集每个点的邻居信息,进行 PageRank 值的收集。然后更新 *PageRank* 值(第 5—10 行),如果某一轮的 PageRank 值与上一轮的插值小于阈值,则跳出迭代。

5.2 Serial Method

对于 CSR 结构,最直观的想法是按照时间顺序进行 PageRank 值计算。如图 2 所示,在 snapshot 处存在一个 CSR 结构来保存快照(CSR-A,如图 2(a)所示),同时将区域 A 的所有 delta 按照入射结点排序后生成另一个 CSR 结构(CSR-B,如图 2(b)所示)。相比 CSR-A,CSR-B 中还存储了增减操作。计算 Page-Rank 时,对于点 A,从 CSR-A 中获取邻居(B,C,D),计算 PageRank。然后访问 CSR-B,补充图 1 中区域 A 的操作带来的改变,例如,如果 CSR-B 中 B 指向 A 的边被删除,就说明之前我们多累加了一个值,将其减去即可。以此类推其他点。



(a)CSR-A 结构 (b)CSR-B 结构

图 2 CSR 结构

Fig. 2 Sstructure of CSR

在算法 3 中,首先判断查询时间是否集中在时间轴的后半部分,若是则利用事先处理好的数据生成 CSR-A(第 2—5 行)。之后不断将 v 添加到 ope_array 中(第 14 行)。如果 time 是目标时间,则将 ope_array 按被指向结点(node2)进行排序,然后生成 CSR-B。根据 CSR-A 和 CSR-B,即可对 Page-Rank 进行计算。

算法 3 线性法 PageRank

```
输入:Historical graph data v[4],half historical graph file h_hg_file,
      target time array t[]
输出:PageRank Value in every target time
1. initialize:int array ope_array[][4] / * 四元组载体 * /
2. if first target time >=half_time then
3.    h_hg_file.rdbuf()
4.    create CSR-A
5. end if
6. while True do
7.    reload v
8.    node1,node2,ope,time←v[0],v[1],v[2],v[3]
9.    if time is one of target time then
10.      sort(ope_array) by node2
11.      create CSR-B
```

```
12.      calPR(CSR-A,CSR-B)
13.  end if
14. ope_array.append(v)
15. end while
```

在算法 4 中，按照与算法 2 的方法初始化 PR[]（第 1 行），之后开始迭代。对于每个点，收集其在 CSR-A 中的邻居，并对 *PageRank* 值进行累加；再根据该点在 CSR-B 中的相关操作进行 *PageRank* 值的更新（第 5－6 行），如果是减边操作则扣除之前多加上的值，如果是加边操作则加上对应的值。当某轮 *PageRank* 值与上一轮的差值小于设定阈值时，结束迭代。

算法 4 链表方法 calPR

```
输入：CSR-A,CSR-B
输出：PageRank Value in the target time
1. initialize：float array PR[]
2. for i=0 up to max_interation do
3.   change=0
4.   for node in nodes do
5.     accumulate rank in CSR-A
6.     update rank according to ope in CSR-B
7.     rank +=damping_value
8.     change +=|PR[node]- rank|
9.     update PR[node] with rank
10.  end for
11.  if change < threshold then
12.    break
13.  end if
14. end for
15. return PR
```

5.3 Parallel Method

维护一个邻居集合来表示某个点出现过的邻居，同时维持一个 *bitset*⟨*query_count*⟩结构来表示在每个目标时间该邻居是否存在，所以如果需要表示所有边，就需要一个长度为边长且元素为 *bitset* 的数组 *nei_exist*[*edge_count*]。假若在第二个目标时间中，CSR 结构中边数组的第 *index* 个边存在，那么 *nei_exist*[*index*][1]=1，反之则为 0。同时，维护两个二维数组 *node_exist*，来记录每个目标时刻中结点的存在状态和出度的信息。比如，在第二个目标时间中如此遍历一遍数据后即可获得所有计算所需信息，不同的计算模块按照分配的任务中的目标时间取出对应时间的邻居，判断该时间对应的标志位上是否为 1，为 1 表示该邻居在该时刻存在，否则不存在。然后就可以计算每个点的 PageRank 值，以此达到并行的效果。

在算法 5 中，如果目标时间都集中在前半部分，那么我们只需读 snapshot 数据，否则需要读取全部数据（第 2－5 行），这些文档都按照被指向结点排序。之后根据 v 中的 Δ 数据来更新 *nei_exist*，同时逐步建立 CSR 结构（第 7－12 行）。根据每个 *target time* 与 *nei_exist* 中相应的状态，即可进行 PageRank 的并行计算（第 13 行）。

算法 5 并行法 PageRank

```
输入：Historical graph data v[4],target time array t[]
输出：PageRank Value in every target time
1. initialize：bitset array nei_exist[]
2. if last target time < half_time then
3.   read front part of the historical file
4. else
5.   read whole part of the historical file
6. end if
7. while True do
8.   reload v
9.   node2,node1,ope,time←v[0],v[1],v[2],v[3]
10.  update nei_exist
11.  Build CSR
12. end while
13. calPR(nei_exist,all target time) in Parallel way
```

5.4 算法总结

Serial Method 中使用的 CSR 结构是利用两个数组来实现的，数组的物理存储是连续的，空间占用也较链表小，但是需要给出数组的长度，以避免空间浪费和溢出。访问数组中的数据会快于链表。在整合构建 CSR 前，需要对已有数据进行排序，时间复杂度为 $O(n\log n)$，之后计算 PageRank 值时只需要在 CSR 中访问数据即可，时间复杂度为 O(1)。

Parallel Method 仅需要扫描一遍按时间及被指向结点排好序的数据，之后按照操作类型判断该边在该时刻是否存在即可，该步骤的时间复杂度为 $O(n)$。在计算某时刻的 PageRank 值时访问 CSR 结构，由于数据已经得到，故获取数据的时间复杂度都为 O(1)。

6 实验

6.1 实验设置

实验数据来自 KONECT，其中记录的是网页互相连接的 Hyperlink networks，实验中所有的数据集都源自该数据集。数据集中每行的数据都是一个四元组（$v1$，$v2$，＋/－，t），表示在 t 时间增加或者删除 $v1$ 指向 $v2$ 的边。表 1 列出了实验参数的设置，表 2 列出了每个数据集的特征。

表 1 实验参数设置

Table 1 Parameters setting

参数	取值范围
数据大小/MB	25，50，100，200，400
精度	10^n，5^n（$n\in[-6,-2]$，$n\in Z$）
目标时间位置	前，后

表 2 数据集信息

Table 2 Detail about dateset

数据集/MB	点数量	边数量	Δ 数量
25	1.11×10^5	6.47×10^5	9.97×10^5
50	2.01×10^5	1.20×10^6	2.00×10^5
100	3.86×10^5	2.30×10^6	4.00×10^5
200	5.86×10^5	4.40×10^6	8.00×10^5
400	8.18×10^5	8.30×10^6	1.59×10^6

实验环境设置如下：CPU 为 16 Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz；核数为 8；内存为 231 022 144 kB，约 220 GB；操作系统为 Linux。

6.2 实验 1：PageRank 迭代轮次实验

由于 PageRank 值需要通过多次迭代才能收敛，因此本

实验探寻不同大小的图的收敛轮次之间存在的关系，以及同一个数据集迭代的轮次和计算收敛的阈值之间的变换关系。

对于每个数据集，控制其计算阈值，记录不同阈值下的迭代轮次，并以阈值为横轴、迭代轮次为纵轴作图。实验结果如图 3 所示。

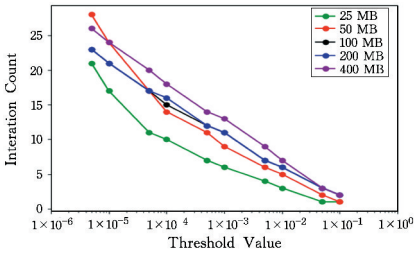


图 3 迭代轮次随阈值的变化

Fig. 3 Iteration needed when threshold changes

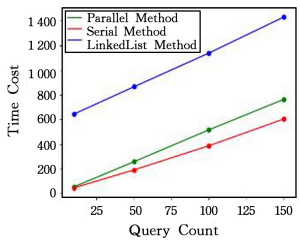
从图 3 可以看出，随着阈值的增大，迭代轮次全部表现为下降趋势，即阈值越小，所需迭代轮次越多；阈值越大，所需迭代轮次越少。

另外，一般数据集越大，相同阈值下所需的迭代次数越多，但是也存在不同数据集在相同阈值下迭代轮次相同或者小的数据集迭代轮次多于大的数据集的情况。

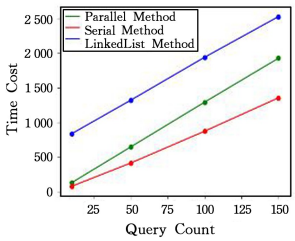
因此可以得出：阈值越大，数据集越大，所需的迭代轮次也就越多。

6.3 实验 2：多参数的性能测试

本实验测试 LinkedList Method，Serial Method 和 Parallel Method 3 种方法随着查询量增多而发生的改变。其中可变的参数包含查询位置、查询跨度。查询位置的前、后分别代表查询集中在时间轴的前半部分和后半部分。实验结果如图 4 所示。



(a)Front Part



(b)Back Part

图 4 400MB 数据集下算法的性能

Fig. 4 Algorithm performance under 400MB dataset

由图 4 可以看出，一般来说，查询时间集中在后半部分时，其计算时间会比集中在前半部分的多，因为集中在后半部分时需要读取 snapshot 数据。

另外，无论数据大小、查询位置，计算时间会随着查询量的增多而增加，而且查询时间和查询数量呈线性关系。

整体来看，其他两种使用 CSR 结构的方法会比使用链表的方法速度更快。其中，LinkedList Method 远远快于链表；而 Parallel Method 在准备数据的过程中可能有所耗时，但是后续计算过程可以并行进行，也会一定程度地提高效率。

6.4 实验 3：单次查询的性能测试

上文之前的测试是在不同跨度、不同查询量的情况下进行的，其每次的测试时间是执行 n 个查询量的总时间。因此我们只能得到计算每个查询的平均时间。

为了更加直观地获知数据集大小和查询时间的关系，我们针对不同数据集，只执行一个查询，即其最大时间戳处的 PageRank 值，以此获得 3 种方法的性能比较，实验结果如图 5 所示。

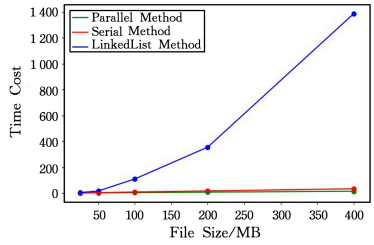


图 5 数据集大小对查询时间的影响

Fig. 5 Impact of dataset size on query time

从图 5 中可以看出，随着数据集的不断增大，使用链表来解决问题的时间将会急剧增加；而使用 CSR 结构来解决问题的 Serial Method 和 Parallel Method 则表现出平稳的态势，二者的性能远远优于使用链表的方法。

另外，为了探究整个过程中时间消耗的主要环节，实验统计了 3 种方法在不同数据集下准备数据的时间与总时间之比（总时间包括准备数据时间和计算时间），结果如图 6 所示。

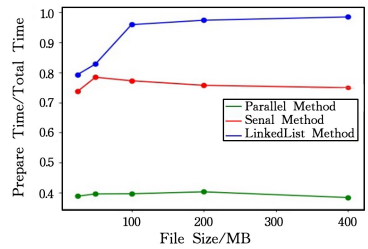


图 6 各方法准备时间的比例

Fig. 6 Proportion of preparation time of different methods

从图 6 可以看出，链表方法的大部分时间都花费在准备数据上，也就是对图结构的维护操作中。在准备数据上，LinkedList Method 花费了总时间的 90%左右，Serial Method 花费了 75%左右，Parallel Method 花费 40%左右。由此可以得出：计算部分的时间并不是总耗时的决定因素，如何高效地维护更新数据才是提高效率的关键。

结束语 本文在历史图的数据上进行 PageRank 问题的相关计算，介绍了历史图的基本特征和 PageRank 的基本思路。在研究主要的解决思路时，考虑到历史图的各种性质（如频繁增删）和处理效率，若使用链表结构，在数据量庞大的情

况下其查询所耗费的时间将会很大,因此本文使用 CSR 结构作为主要的数据结构。在此基础上,借鉴串行计算和并行计算的思想提出了两种不同的方法,并分析了它们各自的特点。实验表明,使用 CSR 结构解决历史图上的 PageRank 问题的性能确实优于使用链表结构的方法,而且效果明显。

目前,在单机情况下,对于处理多个查询,Parallel Method 的效率低于 Serial Method。在接下来的工作中会考虑使用分布式数据处理系统来对 Parallel Method 进行实验优化,尝试提高其计算效率。

参 考 文 献

[1] WU J P,LIN S,XU K,et al. Advances in Evolvable New Generation Internet Architecture[J]. Chinese Journal of Computer,2012,35(6):1094-1108.

[2] PRZYTYCKA T M,SINGH M,SLONIM D K. Toward the dynamic interactome:It's about time[J]. Briefings in Bioinformatics,2010,11(1):15-29.

[3] CAO J. Analysis of Google's PageRank technology[J]. Journal of Information,2002(10):15-18.

[4] CHANG J W,DAI D H. Personalized Recommendation Algorithm Based on PageRank and Spectral Method[J]. Computer Science,2018,45(S2):398-401.

[5] LI C S,LIU X G,JIAO H T,et al. An Improved PageRank Algorithm Based on APP Search System[J]. Computer and Modernization,2018(7):24-27,38.

[6] PEDROCHE F,GARCÍA E,ROMANCE M,et al. On the spectrum of two-layer approach and Multiplex PageRank[J]. Journal of Computational and Applied Mathematics,2018,344:161-172.

[7] HARARY F,GUPTA G. Dynamic graph models[J]. Mathematical and Computer Modelling,1997,25(7):79-87.

[8] DING L L,LI Z D,JI W T,et al. Reachability Query of Large Scale Dynamic Graph Based on Improved Huffman Coding[J]. Acta Electronica Sinica,2017,45(2):359-367.

[9] WANG C,CHEN L. Continuous Subgraph Pattern Search over Graph Streams[C]// International Conference on Data Engineering. IEEE,2009.

[10] FAN W,LI J,LUO J,et al. Incremental graph pattern matching[C]// Acm Sigmod International Conference on Management of Data. ACM,2011.

[11] CHOUDHURY S,HOLDER L,CHIN G,et al. Proceedings of the Workshop on Dynamic Networks Management and Mining[C]// International Conference on Management of Data(SIGMOD/PODS'13). 2013.

[12] WU L K,XIAO X K,DENG D X,et al. Shortest Path and Distance Queries on Road Networks:An Experimental Evaluation[J]. Proceeding of the VLDB Endowment,2012,5(5):406-417.

[13] 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.

[14] WEI Z W,HE X D,X X K,et al. TopPPR:Top-k Personalized PageRank Queries with Precision Guarantees on Large Graphs[C]// International Conference on Management of Data. SIGMOD,2018.

[15] HAVELIWALATH. Topic-sensitive pagerank:A context-sensitive ranking algorithm for web search[J]. IEEE Transactions on Knowledge and Data Engineering,2003,15(4):784-796.

[16] ROUL R K,SAHOO J K,ARORA K. Query-Optimized PageRank:A Novel Approach[M]// Computational Intelligence in Data Mining. Advances in Intelligent Systems and Computing. Singapore:Springer,2019:673-683.

[17] LI C C,KANG Z J,YU H G,et al. Identification Method of Key Nodes in Power System Based on Improved PageRank Algorithm[J]. Transactions of China Electrotechnical Society,2019,34(9):168-175.

[18] LIU Z Y,LI Q F,CENG C,et al. An Optimization Method of PageRank Based on Spark and its Application Research[J]. Journal of CAEIT,2018(4):399-405.

PAN Pei-xian, born in 1996, master candidate. His main research interests include graph data mining and so on.

ZOU Zhao-nian, born in 1979, Ph.D, associate professor, Ph.D supervisor, is a member of China Computer Federation. His main research interests include big data, graph databases and data mining.