DOI:10.13878/j.cnki.jnuist.2016.02.002

彭茂[1]　张媛[1]

带权网络的个性化 PageRank 计算

摘要

PageRank 是衡量网络节点重要性的指标之一,个性化 PageRank 是普通 PageRank 的推广形式.目前关于(个性化)PageRank 的研究主要集中在无权网络,而关于带权网络的研究结果较少.有鉴于此,基于矩阵变换和蒙特卡罗方法,分别给出了在静态和动态带权网络中个性化 PageRank 计算方法,并从理论上分析了算法的性能.实验结果显示,两种算法都优于传统的幂迭代算法.

关键词

PageRank 算法;蒙特卡罗方法;幂迭代法

中图分类号 TP301.6

文献标志码 A

0　引言

挖掘复杂网络的拓扑结构是计算机科学的重要研究课题,而对网络节点做出评级是网络挖掘的内容之一,以 PageRank 为代表的评级系统因其在商业上的巨大成功引起了人们的极大关注.

PageRank[1]是基于投票机制的网络评级系统,从数学上看,PageRank 向量可看成是某一随机游走过程的稳态分布,在此游走过程中,行走者以概率 α 结束当前游走过程,以概率 $1-\alpha$ 从当前所在节点跳跃到邻点,其中 α 称为跳出概率.

关于无权图中的 PageRank 计算目前的研究结果非常丰富,具体可参见 Berkhin[2]、Bianchini 等[3]和 Langville 等[4]的综述报告,以及部分国内学者的应用研究[5-6].但是人们对带权图的 PageRank 计算仍所知甚少,而在很多实际应用中,网络节点间的权重十分重要.比如在一个期刊引用网络[7]中,期刊之间的被引用频次是不一样的,因此,在评价期刊的权威性时,必须考虑边的权重;又如,社交网络中,人与人之间的联系会随着时间的推移而不断加强,因此,研究在动态网络中如何更新 PageRank 向量也是十分有意义的工作.

本文研究带权网络[7-8]中的 PageRank 计算问题,网络可以是静态的,也可以是动态变化的.假设网络顶点数为 n,边数为 m,网络的最大出度为 Δ.在静态带权网络中,本文提出了一个基于 Push 操作的算法,可以在 $O\left(\frac{\Delta}{\alpha}\right)$ 时间内,将计算误差降到常数阶;在动态网络中,本文提出了时间复杂度为 $O\left(\frac{m\Delta}{\alpha^2}\right)$ 和 $O\left(\frac{nf}{\alpha^2}\right)$ 的更新算法,其中 f 为与分布有关且小于 m 的常数.实验结果显示,不论在静态网络还是在动态网络中,本文算法都优于传统的幂迭代算法.

1　预备知识

PageRank 最初定义于无权网络[1],PageRank 向量 $p_{r\alpha}(\boldsymbol{s})$ 可表示为如下线性方程组的唯一解:

$$p_{r\alpha}(\boldsymbol{s}) = \alpha\boldsymbol{s} + (1-\alpha)p_{r\alpha}(\boldsymbol{s})\boldsymbol{W},$$

也可看作是随机游走的稳态分布向量,其中 α 为介于 0 与 1 之间的常数,称为跳出概率,$\boldsymbol{s}$ 为初始向量,$\boldsymbol{W}$ 为状态转移矩阵. 不难看出,PageRank 的初始定义即是 $\boldsymbol{s}$ 为均匀分布时的特例,如果 $\boldsymbol{s}$ 不是均匀分

收稿日期 2014-12-27

资助项目 国家自然科学基金(11401317);南京信息工程大学科研基金(2012X021)

作者简介

彭茂,男,博士,讲师.mpengmath@163.com

1　南京信息工程大学　数学与统计学院,南京,210044

布,则称 $p_{r\alpha}(\boldsymbol{s})$ 为个性化 PageRank[9-10].

对于顶点集为 V、边集为 E 的无权图 $G=(V,E)$,状态转移矩阵 $\boldsymbol{W}$ 通常定义为 $\boldsymbol{W}=\boldsymbol{D}^{-1}\boldsymbol{A}$,其中 $\boldsymbol{D}$ 为各顶点的出度对应的对角阵,$\boldsymbol{A}$ 为图 G 的邻接矩阵.此时,随机游走可看作是从任一顶点按均等概率转移至其邻点.

对于带权图,本文将以上定义做适当修改.记 $\boldsymbol{A}=(a_{ij})_{n\times n}$ 为带权图的权矩阵,顶点数为 n,边数为 m,a_{ij} 为顶点 v_i 与 v_j 之间的权重,若两点之间无边,则其权为零.令 $D_i=\sum_{j=1}^{n}a_{ij}$,$\boldsymbol{D}=\mathrm{diag}(D_1,\cdots,D_n)$.此时,随机游走可视作是从任一顶点按边权重转移至其邻点,状态转移矩阵 $\boldsymbol{W}=\boldsymbol{D}^{-1}\boldsymbol{A}\triangleq(w_{ij})_{n\times n}$,其中 $w_{ij}=a_{ij}/D_i$.

综上所述,无论网络是否带权重,PageRank 向量 $p_{r\alpha}(\boldsymbol{s})$ 都可以表示为前述线性方程组的解.

2 静态带权网络情形

无权图中的 PageRank 向量计算问题目前已被深入研究,幂迭代法是其典型算法:

$$\boldsymbol{p}^{(k)}=\alpha\boldsymbol{s}+(1-\alpha)\boldsymbol{p}^{(k-1)}\boldsymbol{W},$$

算法的误差限通常定义为 $\|\boldsymbol{p}^{(k)}-\boldsymbol{p}^{(k-1)}\|_1\leqslant\varepsilon$.不难看出,幂迭代法的迭代过程至少执行 $\log(\varepsilon)/\log(1-\alpha)$ 次,每一次须计算乘法 $O(m)$ 次,因此时间复杂度可记为 $O(m\cdot\log(\varepsilon)/\log(1-\alpha))$.

本节将给出带权图的一个近似 PageRank 向量的计算方法,其时间复杂度的上界为 $O\left(\frac{\Delta}{\varepsilon\alpha}\right)$,其中 Δ 为图顶点的最大出度.

定义 1 向量 $p_{r\alpha}(\boldsymbol{s}-\boldsymbol{r})$ 称为 PageRank 向量 $p_{r\alpha}(\boldsymbol{s})$ 的 ε- 近似,如果 $\boldsymbol{r}$ 为非负向量,而且对于任意顶点 v 都有 $r(v)\leqslant\varepsilon\cdot\mathrm{outdegree}(v)$.

受到文献[11-12]的启发,本文持续更新一个向量对$(\boldsymbol{p},\boldsymbol{r})$,使之满足满足如下条件:

$$\boldsymbol{p}(\boldsymbol{I}-(1-\alpha)\boldsymbol{W})=\alpha(\boldsymbol{s}-\boldsymbol{r}).$$

在算法的开始阶段 $\boldsymbol{p}=0,\boldsymbol{r}=\boldsymbol{s}$,向量 $\boldsymbol{p}$ 和 $\boldsymbol{r}$ 通过如下 Push 过程更新:

Let $\boldsymbol{p}'=\boldsymbol{p}$ and $\boldsymbol{r}'=\boldsymbol{r}$ except for these changes:
1. $\boldsymbol{p}'(u)=\boldsymbol{p}(u)+\alpha\cdot\beta\cdot\boldsymbol{r}(u)$,
2. $\boldsymbol{r}'(u)=\boldsymbol{r}(u)-\beta\cdot\boldsymbol{r}(u)+(1-\alpha)w_{u,u}\cdot\beta\cdot\boldsymbol{r}(u)$,
3. $\boldsymbol{r}'(v)=\boldsymbol{r}(v)+(1-\alpha)w_{u,v}\cdot\beta\cdot\boldsymbol{r}(u)$ for each v such that $(u,v)\in E$

此处,如果状态转移矩阵被定义为 $\boldsymbol{W}=\boldsymbol{D}^{-1}\boldsymbol{A}\triangleq(w_{ij})_{n\times n}$,则 Push 过程中的 $w_{u,u}$ 可简化为 0;β 为预设的介于 0 和 1 之间的常数,若无特别说明,本文记 $\beta=1/2$.

容易看出,向量对$(\boldsymbol{p},\boldsymbol{r})$ 的定义式方程等价于下式:

$$\boldsymbol{r}=\boldsymbol{s}-\boldsymbol{p}\left(\frac{1}{\alpha}\boldsymbol{I}-\frac{1-\alpha}{\alpha}\boldsymbol{W}\right),$$

从基础矩阵乘法可得如下引理:

引理 1 设 $\boldsymbol{p}'$ 和 $\boldsymbol{r}'$ 为向量 $\boldsymbol{p},\boldsymbol{r}$ 经过 Push 过程之后所得向量,则

$$\boldsymbol{p}=p_{r\alpha}(\boldsymbol{s}-\boldsymbol{r})\Rightarrow\boldsymbol{p}'=p_{r\alpha}(\boldsymbol{s}-\boldsymbol{r}').$$

在执行 Push 的过程中,向量 $\boldsymbol{r}$ 的向量和持续减少,而 $\boldsymbol{p}$ 的向量和持续增加,增加量和减少量相同,因此可认为向量 $\boldsymbol{r}$ 中的某些概率转移到了向量 $\boldsymbol{p}$ 中.对所有满足 $r(u)\geqslant\varepsilon\cdot\mathrm{outdegree}(u)$ 的顶点执行 Push 操作,则得到如下计算 ε- 近似 PageRank 向量的算法:

ApxPR($\boldsymbol{s},\alpha,\varepsilon$):
1. Let $\boldsymbol{p}=0$ and $\boldsymbol{r}=\boldsymbol{s}$
2. While $\boldsymbol{r}(u)>\varepsilon\cdot\mathrm{outdegree}(u)$ for some vertex u
(a) Pick any vertex u where $\boldsymbol{r}(u)>\varepsilon\cdot\mathrm{outdegree}(u)$,
(b) Apply Push(u),
3. Return $\boldsymbol{p}$ and $\boldsymbol{r}$.

定理 1 若 $\|\boldsymbol{s}\|_1\leqslant1$ 而且 $\varepsilon\in(0,1]$,则算法 ApxPR($\boldsymbol{s},\alpha,\varepsilon$) 返回 $p_{r\alpha}(\boldsymbol{s})$ 的 ε- 近似 PageRank 向量 $\boldsymbol{p}$,并且时间复杂度上界为 $O\left(\frac{1}{\varepsilon\alpha}\right)$.

证明 因为 Push 操作不改变向量对$(\boldsymbol{p},\boldsymbol{r})$ 满足 $\boldsymbol{p}=p_{r\alpha}(\boldsymbol{s}-\boldsymbol{r})$ 的性质,而算法的停止条件又使得每一个顶点 u 均有 $\boldsymbol{r}(u)\leqslant\varepsilon\cdot\mathrm{outdegree}(u)$ 成立,所以算法 ApxPR($\boldsymbol{s},\alpha,\varepsilon$) 返回的向量 $\boldsymbol{p}$ 是 PageRank 向量 $p_{r\alpha}(\boldsymbol{s})$ 的 ε- 近似.

设算法 APXPR 执行 Push 操作的次数为 T.当进行第 i 次 Push 操作时,顶点 u 对应的 $\boldsymbol{r}$ 向量的分量至少为 $\varepsilon\cdot\mathrm{outdegree}(u)$,而 $\|\boldsymbol{r}\|_1$ 的减少量最少为 $\varepsilon\cdot\mathrm{outdegree}(u)\cdot\alpha/2$(此时 $\beta=1/2$).又因为 $\|\boldsymbol{s}\|_1\leqslant1$,所以

$$\sum_{i=1}^{T}\mathrm{outdegree}(u_i)\leqslant\frac{2}{\varepsilon\alpha},$$

证毕.

注意到,本文采用的误差测度为 $\boldsymbol{r}(u)\leqslant\varepsilon\cdot\mathrm{outdegree}(u)$,而通常衡量 PageRank 计算的误差标准为 $\|\boldsymbol{r}\|_1\leqslant\varepsilon$,因此将上式的 outdegree($u$) 放大

为图的最大出度 Δ, 则 APXPR 算法的执行时间为 $O\left(\frac{1}{\varepsilon\alpha}\right)$.

与经典的幂迭代法 $\boldsymbol{p}^{(k)}=\alpha\boldsymbol{s}+(1-\alpha)\boldsymbol{p}^{(k-1)}\boldsymbol{W}$ 比较,幂迭代法的时间复杂度为 $O\left(\frac{m\log\varepsilon}{\log(1-\varepsilon)}\right)$,远比 APXPR 算法的时间复杂度 $O\left(\frac{\Delta}{\varepsilon\alpha}\right)$ 要高.

3 动态带权网络情形

现实网络的拓扑结构通常是不断变化的,例如在社交网络中,会有新人的加入、朋友之间的联系加强等情况.如果每一次网络的更新都重新计算 PageRank,那么所耗成本是很高的,因此,需要以已有的 PageRank 为基础,设计快的更新算法.本节仅考虑有向网络中边权重发生改变的情况.在现实网络中,可能在同一时间有大量更新在进行,为便于比较,本文假设边权重的改变依次进行,即每一次只改变一条边的权重.

3.1 局部近似算法

令 $\boldsymbol{A}=(a_{ij})_{n\times n}$ 为有向带权图的权矩阵,顶点数为 n,边数为 m,记 $D_i=\sum_{j=1}^{n}a_{ij}$,$\boldsymbol{D}=\mathrm{diag}(D_1,\cdots,D_n)$. 此时,状态转移矩阵 $\boldsymbol{W}=\boldsymbol{D}^{-1}\boldsymbol{A}\triangleq(w_{ij})_{n\times n}$.为简便计,假设边 (v_i,v_j) 的权重改变量为 $\Delta_{ij}=tD_i$,记更新后的状态转移矩阵为 $\widetilde{\boldsymbol{W}}=(\tilde{w}_{ij})_{n\times n}$.

$$\begin{cases}\tilde{w}_{ij}=\dfrac{\tilde{a}_{ij}}{\sum_{j=1}^{n}\tilde{a}_{ij}}=\dfrac{\tilde{a}_{ij}}{\sum_{k\neq j}\tilde{a}_{ik}+a_{ij}}=\dfrac{a_{ij}+tD_i}{\sum_{j=1}^{n}a_{ij}+tD_i}=\\ \qquad\dfrac{a_{ij}+tD_i}{(1+t)D_i}=\dfrac{1}{1+t}(w_{ij}+t),\\ \tilde{w}_{ik}=\dfrac{\tilde{a}_{ik}}{\sum_{j=1}^{n}\tilde{a}_{ij}}=\dfrac{a_{ik}}{\sum_{j=1}^{n}a_{ij}+tD_i}=\dfrac{1}{1+t}\dfrac{a_{ik}}{D_i}=\\ \qquad\dfrac{w_{ik}}{1+t}\ (k\neq j).\end{cases}$$

$$\begin{cases}\tilde{w}_{ij}-w_{ij}=\dfrac{1}{1+t}(w_{ij}+t)-w_{ij}=\dfrac{t}{1+t}(1-w_{ij}),\\ \tilde{w}_{ik}-w_{ik}=\dfrac{w_{ik}}{1+t}-w_{ik}=-\dfrac{t}{1+t}w_{ik}.\end{cases}$$

设 $\boldsymbol{p}$ 为算法 ApxPR 返回的 ε- 近似 PageRank 向量,本文将 $\boldsymbol{p}$ 作为计算新 PageRank 向量的初始向量,使得更新后得到的 PageRank 向量满足:

$$\tilde{\boldsymbol{p}}\left(\frac{1}{\alpha}\boldsymbol{I}-\frac{1-\alpha}{\alpha}\widetilde{\boldsymbol{W}}\right)=\boldsymbol{s}-\tilde{\boldsymbol{r}}.$$

将 $\boldsymbol{W}$ 改变为 $\widetilde{\boldsymbol{W}}$ 后,

$$\begin{cases}\tilde{\boldsymbol{p}}=\boldsymbol{p},\\ \tilde{r}(i)=r(i),\\ \tilde{r}(j)=r(j)+\dfrac{1-\alpha}{\alpha}(w_{ij}-w_{ij})=\\ \qquad r(j)+\dfrac{1-\alpha}{\alpha}\dfrac{t}{1+t}p(i)(1-w_{ij}),\\ \tilde{r}(k)=r(k)+\dfrac{1-\alpha}{\alpha}(w_{ik}-w_{ik})=\\ \qquad r(k)-\dfrac{1-\alpha}{\alpha}\dfrac{t}{1+t}p(i)w_{ik}(k\neq j).\end{cases}$$

如果 $t>0$,那么 $r(j)$ 会增加 $\frac{1-\alpha}{\alpha}\frac{t}{1+t}p(i)\cdot(1-w_{ij})$,而 $r(k)$ 会减少 $\frac{1-\alpha}{\alpha}\frac{t}{1+t}p(i)w_{ik}$.因此,有必要检查 $\tilde{\boldsymbol{p}}$ 是否仍为 ε- 近似 PageRank 向量.

1) 如果 $r\geqslant 0$,而且任意顶点 v 都满足 $r(v)\leqslant\varepsilon\cdot\mathrm{outdegree}(v)$,则 $\tilde{p}$ 仍为 ε- 近似 PageRank 向量,此时无需进一步计算.

2) 如果 $r\geqslant 0$,但是某些顶点 v 使得 $r(v)>\varepsilon\cdot\mathrm{outdegree}(v)$,则最多会有概率 $\frac{1-\alpha}{\alpha}\frac{t}{1+t}p(i)w_{ik}$ 传递到 $\tilde{\boldsymbol{p}}$,假设更新算法执行 Push 操作的次数为 T,类似于前述定理的分析,有

$$\varepsilon\frac{\alpha}{2}\sum_{k=1}^{T}\mathrm{outdegree}(u_k)\leqslant\frac{1-\alpha}{\alpha}\frac{t}{1+t}p(i)(1-w_{ij}),$$

$$\sum_{k=1}^{T}\mathrm{outdegree}(u_k)\leqslant\frac{2}{\varepsilon\alpha^2}\frac{(1-\alpha)t}{1+t}p(i)(1-w_{ij})\leqslant\frac{2}{\varepsilon\alpha^2}.$$

为了将误差界降为 $\|\tilde{\boldsymbol{r}}\|_1\leqslant\varepsilon$,新增加的时间不超过 $O\left(\frac{\Delta}{\varepsilon\alpha^2}\right)$,其中 Δ 为图的最大出度.

3) 如果某些顶点 v 使得 $r(v)<0$,则 $\tilde{\boldsymbol{p}}$ 不适合作为更新的初始向量,我们须直接以 ApxPR 算法计算此时网络的 ε- 近似 PageRank 向量,时间复杂度为 $O\left(\frac{\Delta}{\varepsilon\alpha}\right)$.

UApxPR($\boldsymbol{s},\alpha,\varepsilon$):

1. Let $\tilde{\boldsymbol{p}}=\boldsymbol{p}$ and $\tilde{\boldsymbol{r}}=\boldsymbol{s}-\tilde{\boldsymbol{p}}\left(\frac{1}{\alpha}\boldsymbol{I}-\frac{1-\alpha}{\alpha}\widetilde{\boldsymbol{W}}\right)$
2. If $\boldsymbol{r}(u)<0$ for some vertex u:
(a) Run Algorithm ApxPR ($\boldsymbol{s},\alpha,\varepsilon$)
(b) Return $\tilde{\boldsymbol{p}}$ and $\tilde{\boldsymbol{r}}$.

3. $r(u) > \varepsilon \cdot \text{outdegree}(u)u$ for some vertex u,
(a) Pick any vertex u where $r(u) > \varepsilon \cdot$ outdegree(u),
(b) Apply Push (u),
4. Retrn $\tilde{\boldsymbol{p}}$ and $\tilde{\boldsymbol{r}}$.

定理 2 更新网络的一条边带来的额外 PageRank 计算时间为 $O\left(\frac{\Delta}{\varepsilon\alpha^2}\right)$.

因为 $O\left(\frac{\Delta}{\varepsilon\alpha}\right)$ 明显小于 $O\left(\frac{\Delta}{\varepsilon\alpha^2}\right)$,所以定理显然成立,而如果需要更新 m 条边,所需的时间则为 $O\left(\frac{m\Delta}{\varepsilon\alpha^2}\right)$. 另外,注意到在实际网络中 $\frac{1-\alpha}{\alpha} \cdot \frac{t}{1+t}p(i)(1-w_{ij})$ 远小于1,所以定理的时间估计仅为了考虑理论的紧性,实际网络的计算所需时间会小得多.

3.2 蒙特卡罗方法

蒙特卡罗方法也称统计模拟方法,是以概率统计理论为指导的一类实用算法,而 PageRank 向量的定义方程式:

$$p_{r\alpha}(\boldsymbol{s}) = \alpha\boldsymbol{s} + (1-\alpha)p_{r\alpha}(\boldsymbol{s})\boldsymbol{W}$$

具备随机游走的数学特征,因此本文采用蒙特卡罗策略[13-14]来计算 PageRank 向量. 本节先考虑 $\boldsymbol{s}=(1/n,1/n,\cdots,1/n)$ 的情形,也即普通 PageRank 的计算.

对于普通的 PageRank 向量计算,本文在网络的每个顶点进行 R 次如下的随机游走:每一步它都以概率 α 结束游走,而以概率 $1-\alpha$ 走到某个邻点,邻点的选择概率正比于对应的邻边的权重.不难看出,此时马尔科夫链的链长的期望值为 $1/\alpha$.

对每一个顶点 v,令 X_v 为所有顶点的 R 次随机游走中顶点 v 的出现次数,则顶点 v 的 PageRank 值可近似为

$$\tilde{\pi}_v = \frac{X_v}{nR/\alpha}.$$

当网络结构发生改变时,考虑更新马尔科夫链来更新各点的 PageRank 值.为此,本文假设网络的更新模式为各条边以在线(On-line)的方式添加到图中.

定理3 假设在时间 t 时刻$(1 \leqslant t \leqslant m)$,边$(u_t,v_t)$ 被加入到图中,边(u_t,v_t) 的权重为 w_t.记 M_t 为需要更新的马尔科夫链的条数,则

$$E[M_t] \leqslant \frac{nR}{\alpha}\frac{w_t}{w_1+\cdots+w_t}.$$

证明 一条链需要更新仅当此链经过 u_t 并且以 v_t 为后继点.因为链长的期望值为 $1/\alpha$,所以每条链中出现 u_t 的次数的期望值为 π_{u_t}/α.在 u_t 处选择 v_t 为后继的概率为 $w_t/a(u_t)$,其中 $a(u_t)$ 为顶点 u_t 处的边权重之和.那么

$$E[M_t] \leqslant \sum_u \frac{nR}{\alpha}\pi_u Pr[u_t=u]\sum_v \frac{w_t}{a(u)}Pr[v_t=v \mid u_t=u] = \frac{nRw_t}{\alpha}\sum_u \frac{\pi_u}{a(u)}Pr[u_t=u] = \frac{nRw_t}{\alpha}E\left[\frac{\pi_{u_t}}{a(u_t)}\right],$$

$E\left[\frac{\pi_{u_t}}{a(u_t)}\right]$ 的取值与网络的增长模型有关.此处,假设各边完全以随机方式加入,则此时有 $Pr[u_t=u] = \frac{a(u)}{w_1+\cdots+w_t}$,则

$$E[M_t] \leqslant \frac{nRw_t}{\alpha}E\left[\frac{\pi_{u_t}}{a(u_t)}\right] = \frac{nRw_t}{\alpha}\sum_u \frac{\pi_u}{a(u)}Pr[u_t=u] = \frac{nRw_t}{\alpha}\sum_u \frac{\pi_u}{a(u)}\frac{a(u)}{w_1+\cdots+w_t} = \frac{nR}{\alpha}\frac{w_t}{w_1+\cdots+w_t}\sum_u \pi_u = \frac{nR}{\alpha}\frac{w_t}{w_1+\cdots+w_t}.$$

为了对整体的所用时间给出上界,本文给出如下引理:

引理2 设 $0 < a_1 \leqslant a_2 \leqslant \cdots \leqslant a_m$,$w_1,w_2,\cdots,w_m$ 为 $a_1,a_2,\cdots,a_m$ 的排列,则

$$f(w_1,\cdots,w_m) = \frac{w_1}{w_1} + \frac{w_2}{w_1+w_2} + \cdots + \frac{w_m}{w_1+\cdots+w_m}$$

取得最大值仅当 $w_i = a_i$ 对任意 i 成立.

证明 对 $a_1,a_2,\cdots,a_m$ 的任一排列 $w_1,w_2,\cdots,w_m$,

1) 若 $w_i > w_{i+1}$,因为

$$\frac{w_i}{w_1+\cdots+w_{i-1}+w_i} + \frac{w_{i+1}}{w_1+\cdots+w_{i-1}+w_i+w_{i+1}} < \frac{w_{i+1}}{w_1+\cdots+w_{i-1}+w_{i+1}} + \frac{w_i}{w_1+\cdots+w_{i-1}+w_{i+1}+w_i},$$

所以 $f(w_1,\cdots,w_{i-1},w_i,w_{i+1},\cdots,w_m) < f(w_1,\cdots,w_{i-1},w_{i+1},w_i,\cdots,w_m)$.

2) 若 $w_i > w_j$,重复执行相邻位的互换,可得:$f(w_1,\cdots,w_i,\cdots,w_j,\cdots,w_m) < f(w_1,\cdots,w_j,\cdots,w_i,\cdots,w_m)$.

所以,引理得证.

f 的最大值与 $a_1,a_2,\cdots,a_m$ 的取值有关,f 显然有上界 m.但在更多的实际网络中,f 的最大值是可以改

进的.例如,在一般的无权图中,$a_1 = a_2 = \cdots = a_m = 1$,因为 $\sum_{t=1}^{m} 1/t \approx \log(m)$,所以 f 的上界为 $\log(m)$.

定理 4 对于一个有 n 个顶点、m 条边的图,保持 m 条边持续更新所花时间为 $O\left(\frac{nRf}{\alpha^2}\right)$,其中 R 为每个顶点的马尔科夫链的条数,f 为介于 1 与 m 之间的与网络模型有关的常数.

证明

$$E[M_t] \leqslant \frac{nR}{\alpha} E\left[\frac{\pi_{u_t} a(u_t,v_t)}{a(u_t)}\right] = \frac{nR}{\alpha} \frac{w_t}{w_1 + \cdots + w_t}.$$

对于每一个需要更新的链,可以在跳转点 u_t 处重做游走,或者直接在链的初始位置重做游走.平均地,每一条链需要跳转 $1/\alpha$ 次(等于链的平均长度),从上述引理可得:

$$\frac{nR}{\alpha^2} \sum_{t=1}^{m} \frac{w_t}{w_1 + \cdots + w_t} \leqslant \frac{nRf}{\alpha^2},$$

证毕.

以上为 $\boldsymbol{s} = (1/n, 1/n, \cdots, 1/n)$ 时,也即计算普通 PageRank 的结果.如果是计算个性化 PageRank,不妨记 $p_{r\alpha}(v)$ 为初始向量 $\boldsymbol{s}$ 对应为图中的顶点 v 的情况,即在向量 $\boldsymbol{s}$ 中,对应顶点 v 的位置取值为 1,其余处取值为 0.设 $\boldsymbol{s} = (s_1, s_2, \cdots, s_n)$,则

$$p_{r\alpha}(\boldsymbol{s})(\boldsymbol{I} - (1-\alpha)\boldsymbol{W}) = \alpha \boldsymbol{s},$$

不难看出

$$p_{r\alpha}(\boldsymbol{s}) = \sum_{i=1}^{n} s_i p_{r\alpha}(\boldsymbol{v}_i),$$

因此可以先计算出各点的 $p_{r\alpha}(v)$,进而得到需要的 $p_{r\alpha}(\boldsymbol{s})$ 向量.为此,可以在顶点 v 处做若干次随机游走,然后统计各顶点出现的频率即可,分析与普通 PageRank 类似,具体可参考文献[14-15].

4 数值实验

本节将对幂迭代法、Push 算法和蒙特卡罗算法在不同网络中进行性能测试.

在试验中,网络以如下方式生成:假设网络为无向网络;顶点按在线方式依次抵达;当新顶点到达之后,新顶点以固定概率与所有已知点连边;新顶点必须与至少一个已知点连边以满足网络连通性;边权重在区间(0,1)之间均匀随机选取.

因为幂迭代法的误差界设定为 $\|\boldsymbol{p}^{(k)} - \boldsymbol{p}^{(k-1)}\| \leqslant \varepsilon$,而 Push 算法的误差设定为 $r(v) \leqslant \varepsilon \cdot \text{outdegree}(v)$,为测试公平,本文将 Push 算法的误差标准设定为 $\|\boldsymbol{r}\|_1 \leqslant \varepsilon$.

在所有的比较中,跳出概率 $\alpha = 0.15$,误差限 $\varepsilon = 1 \times 10^{-6}$,Push 算法的参数 β 设为 1.实验在不同大小(Cardinality)的图和不同稠密度(Density)的图中进行,同样的大小和稠密度参数下,随机生成 10 个图测试算法的平均执行时间.程序以 C 语言编程,并运行在内存 2 GB、CPU 主频 2.13 GHz 的计算机上,操作系统 Win7(64).

关于静态图中的 PageRank 计算,图 1 显示了稠密度为 0.1 时不同算法的计算时间,图 2 显示了顶点数为 5 000 时不同稠密度网络的 PageRank 计算时间.结果显示,在低稠密度时,Push 算法和蒙特卡罗算法有类似的计算效率,而且都比幂迭代法速度更快.从图 2 观察可知,幂迭代法和蒙特卡罗方法对图的抽密度不敏感,而 Push 算法在中抽密度(0.6)时

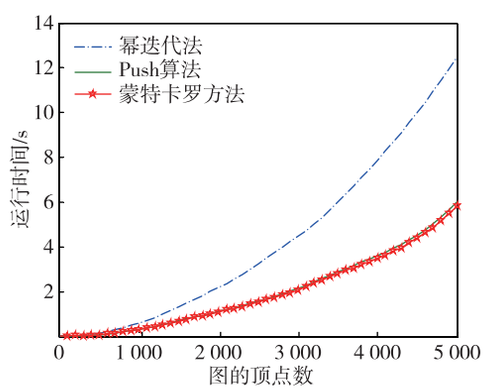

图 1 稠密度为 0.1 时不同算法的计算时间

Fig. 1 Comparison of computation time consumed by three algorithms when PageRank: density equals 0.1 for static web

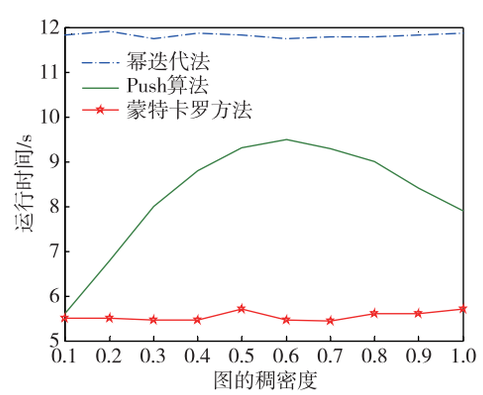

图 2 顶点数量为 5 000 时不同算法的计算时间

Fig. 2 Comparison of computation time consumed by three algorithms when PageRank: cardinality equals 5 000 for static web

速度较慢,但即便如此,蒙特卡罗方法和 Push 算法的计算效率也都优于幂迭代法.

关于动态图中的 PageRank 计算,试验也在不同大小和不同稠密度的图中进行,在试验中,将图中的某一条边做加权处理.因为幂迭代法在更新 PageRank 时通常只是简单地重新计算一次,其执行时间已在图 1 和图 2 中得到体现,因此只比较蒙特卡罗方法和 Push 算法.图 3 和图 4 的结果显示,在图的更新量较少时,Push 算法要优于蒙特卡罗方法.

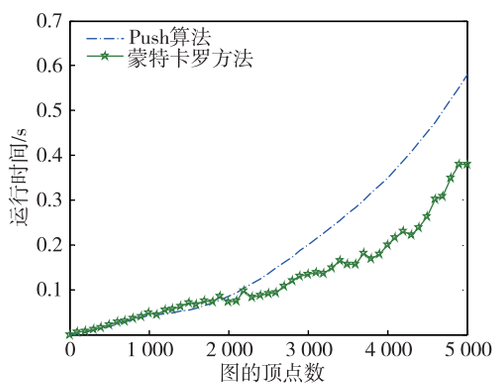

图 3 稠密度为 0.1 时不同更新算法的计算时间

Fig. 3 Comparison of computation time consumed by two algorithms when PageRank: density equals 0.1 for dynamic web

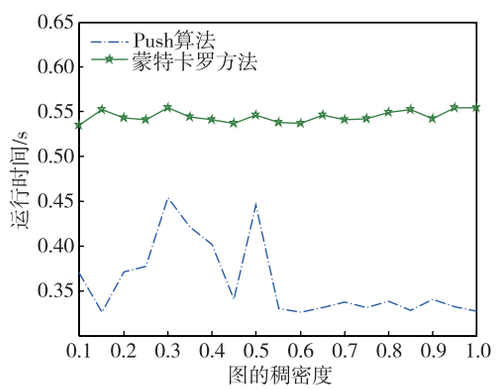

图 4 顶点数量为 5 000 时不同更新算法的计算时间

Fig. 4 Comparison of computation time consumed by two algorithms when PageRank: cardinality equals 5 000 for dynamic web

5 结论

PageRank 向量在复杂网络研究中占有重要地位,在过去的 10 余年中,关于无权图中的 PageRank 计算其研究结果已很多,但是在带权图中的研究结果却远为丰富,而事实上,带权网络反而是更易见到的网络类型.本文讨论带权网络中的 PageRank 计算问题,对于静态网络和动态网络,在矩阵变换和蒙特卡罗方法的基础上,提出了个性化 PageRank 计算的近似方法.数值试验显示,本文算法都优于传统的幂迭代算法.

参考文献

References

[1] Page L,Brin S,Motwani R,et al.The PageRank citation ranking: Bringing order to the web [R]. Technical Report,Stanford InfoLab,1999

[2] Berkhin P. A survey on PageRank computing [J]. Internet Mathematics,2005,2(1):73-120

[3] Bianchini M,Gori M,Scarselli F.Inside PageRank [J]. ACM Transactions on Internet Technology (TOIT),2005,5(1):92-128

[4] Langville A N,Meyer C D.Deeper inside PageRank [J]. Internet Mathematics,2004,1(3):335-380

[5] 宋聚平,王永成,尹中航,等.对网页 PageRank 算法的改进[J].上海交通大学学报,2003,37(3):397-400
SONG Juping, WANG Yongcheng, YIN Zhonghang, et al.Improved PageRank algorithm for ordering web pages [J].Journal of Shanghai Jiaotong University,2003,37(3):397-400

[6] 黄德才,戚华春.PageRank 算法研究[J].计算机工程,2006,32(4):145-146
HUANG Decai, QI Huachun. PageRank algorithm research [J]. Computer Engineering, 2006, 32(4): 145-146

[7] Bollen J,Rodriquez M A,Sompel H.Journal status [J]. Scientometrics,2006,69(3):669-687

[8] Xing W P,Ghorbani A.Weighted PageRank algorithm [C]// Second Annual Conference on Communication Networks and Services Research (CNSR'04),2004:305-314

[9] Jeh G,Widom J.Scaling personalized web search [C]// Proceedings of the 12th International Conference on World Wide Web,2003:271-279

[10] Csalogány K,Fogaras D,Rácz B,et al.Towards scaling fully personalized PageRank:Algorithms,lower bounds, and experiments [J]. Internet Mathematics, 2005, 2(3):333-358

[11] Andersen R,Chung F,Lang K.Local graph partitioning using PageRank vectors [C]// Proceedings of the 47th Annual IEEE Symposium on Foundation of Computer Science,2006:475-486

[12] Graham F,Tsiatas A.Finding and visualizing graph clusters using PageRank optimization algorithms and models for the web-graph [J].Lecture Notes in Computer Science,2010,6516:86-97

[13] Avrachenkov K, Litvak N, Nemirovsky D, et al. Monte Carlo methods in PageRank computation:When one iteration is sufficient [J].SIAM Journal on Numerical Analysis,2008,45(2):890-904

[14] Bahmani B,Chowdhury A,Goel A.Fast incremental and personalized PageRank[J].Proceedings of VLDB Endowment,2010,4(3):173-184
[15] Sarma A D,Gollapudi S,Panigrahy R.Estimating PageRank on graph streams[C]// Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems,2008:9-12

Computing personalized PageRank in weighted networks

PENG Mao[1] ZHANG Yuan[1]
1 School of Mathematics & Statistics,Nanjing University of Information Science & Technology,Nanjing 210044

Abstract PageRank assigns authority weights to each web page based on the web hyperlink structure,while the personalized PageRank is a generalized version of ordinary PageRank.The computation of personalized PageRank vector in unweighted web is well studied in the past decades,but little is known for the case of weighted webs.In this paper,we analyze the algorithms for PageRank computations in static as well as dynamic weighted networks.The algorithms are based on matrix transformation or Monte Carlo methods,and are analyzed theoretically for computation performance.Experiments show that the proposed localized algorithm outperforms power iteration and a referenced Monte Carlo method.

Key words PageRank algorithm;Monte Carlo method;power iteration