计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20160210
53(8):1806-1818,2016

IncPR:一种基于增量计算的并行 PageRank 算法

姜双双　廖　群　杨愚鲁　李　涛
(南开大学计算机与控制工程学院　天津　300350)
(highfly@mail.nankai.edu.cn)

IncPR: An Incremental Parallel PageRank Algorithm

Jiang Shuangshuang, Liao Qun, Yang Yulu, and Li Tao
(*College of Computer and Control Engineering, Nankai University, Tianjin* 300350)

Abstract　PageRank algorithm plays an important role in various areas such as information retrieval and social network analysis. The continuously increasing size of networks and the timeliness requirements make it bring enormous computational overhead to repeat calculating PageRank for a massive and evolving network. Therefore, IncPR, a PageRank algorithm based on the idea of incremental computation model is proposed. IncPR reuses existing results from previous computation and modifies PageRank values according to the changed part of data sets accumulatively via an extended Monte Carlo method, which can effectively avoid reduplicated computation and improve the performance of PageRank computing with no additional storage overhead. Theoretical analysis shows that the computational complexity of IncPR processing an evolving network with m nodes and n edges changed is $O((cm+n)R/c^2)$, where c is the reset probability and R is the number of random walks starting from each node. The computational complexity of IncPR is lower than other existing related algorithms. Our evaluations with typical real-world graphs upon a Hama cluster also demonstrate that IncPR obtains PageRank values with equal (or even higher) accuracy as the baseline Monte Carlo PageRank algorithm and reduces the amount of computation significantly. In experiments with 0.01% data changed, IncPR reduces over 99.9% amount of computation.

Key words　PageRank; Web mining; incremental computing; Monte Carlo algorithm; parallel and distributed processing

摘　要　广泛的互联网的商业应用使 PageRank 算法有重要地位.网络规模不断地增大,同时网络变化带来的时效性要求,也使 PageRank 计算对计算资源的要求不断地提高.为降低该问题对计算资源的消耗水平,降低计算成本,一种基于增量计算思想的 PageRank 算法:IncPR 被提出.IncPR 通过重用已有的结果,增量地获得数据变化后的结果.该算法在并行计算环境中,能够有效地降低计算量,缩短计算时间.理论分析表明,该算法计算结果的误差范围与蒙特卡罗 PageRank 算法相当,其时间复杂度优于其他已有的相关算法,且不引入额外的存储开销.在分布式集群 Hama 上进行的实验验证了理论分析的结果,IncPR 在得到与蒙特卡罗 PageRank 算法同等(甚至更高)结果精度的情况下,显著地降低了计算量.

关键词　PageRank;Web 数据挖掘;增量计算;蒙特卡罗算法;并行与分布式处理

中图法分类号　TP391

收稿日期:2016-03-21;**修回日期**:2016-05-25

随着电子商务、物联网、社交网络、生物信息学等诸多领域的蓬勃发展,大量有价值的数据快速地产生并积累,“大数据”的计算处理成为近年来的研究热点之一.大量的统计结果显示,大数据应用问题普遍具有数据的总量大且增长速度快的特点[1].截止到 2016 年 3 月,Google 抓取的网页已超过 60 万亿,并以每天超过 450 亿个网页的速度增长[2-3];FaceBook 的用户数量已达到 13 亿并保持每秒 5 位用户的增长速度[4].当数据发生变化时,需要对数据重新处理以更新结果,因此,高效地处理频繁变化的大数据具有重大意义.

PageRank[5]算法在搜索引擎、信息检索、社交网络挖掘等多个领域得到广泛的应用,具有重要地位.PageRank 算法所处理的互联网络和社交网络数据是典型的频繁变化的大数据集,具有总量大、更新速度快、每次更新的数据占总量比例小的特点.为满足结果的时效性要求,只要数据更新就需要重计算以更新结果.在频繁变化的大数据集上反复地进行 PageRank 计算,会消耗巨大的计算资源,计算成本高.

尽管以 MapReduce[6],Spark[7]及 Pregel[8]集群等为代表的分布式计算技术、以及很多并行 PageRank 算法[9-11]能够提高 PageRank 的计算速度.但这些技术仍不能有效地降低反复计算带来的高资源消耗和高计算成本.

因此针对这一问题本文通过扩展一种典型的 PageRank 计算算法——蒙特卡罗算法[12],提出了一种基于增量计算的 PageRank 算法——IncPR.当数据集发生变化时,IncPR 利用的是已有的 PageRank 计算结果,不需要保存其他历史数据,没有引入额外的存储开销;在进行处理时,IncPR 根据数据的变化情况在已有的计算结果上增量地进行更新,获得新数据集的结果,这有效地降低了计算量,提高了计算效率.此外,IncPR 能高效处理网页增加、减少、网页链接关系变更等多种情况,这也是其较之于部分相关增量算法的优势之一.

理论分析证明,IncPR 得到结果的误差范围和蒙特卡罗 PageRank 算法得到的结果误差范围的量级相当,即 IncPR 具有与蒙特卡罗 PageRank 算法相同的正确性.在增加 m 个节点和 n 条边的情况下,IncPR 的时间复杂度为 $O((cm+n)R/c^2)$,其中参数 c 和 R 分别代表蒙特卡罗方法的停止概率和启动次数.IncPR 的时间复杂度优于其他已有的增量 PageRank 算法.在分布式集群 Hama[13]上进行的实验验证了理论分析的结果,实验数据表明在得到与蒙特卡罗 PageRank 算法同等(甚至更高)精度结果的情况下,IncPR 显著地降低了计算量.

综上,本文工作的贡献有 3 点:

1) 提出了一种基于增量计算的 PageRank 算法 IncPR.通过在已有结果的基础上,增量地更新得到新结果的方法,IncPR 能有效地降低 PageRank 的计算量,且不引入额外的存储开销;

2) 通过理论分析证明了 IncPR 与蒙特卡罗 PageRank 算法的结果具有相等量级的精度,且时间复杂度优于已有的增量 PageRank 算法;

3) 在 Hama 集群上基于大规模真实图数据的大量实验,验证了 IncPR 的正确性和计算效率.IncPR 在保证结果正确的前提下能有效降低 PageRank 的计算量.

1 相关工作

1.1 PageRank 算法

PageRank 定义了一种网络中节点重要性的度量指标.PageRank 算法使用马尔可夫链为用户浏览网页的行为建模,进而将 PageRank 值定义为以全体网页的集合为状态空间的马尔可夫链的稳态分布.对于包含 n 个网页的网络,其转移矩阵 $\boldsymbol{P}$ 是一个 $n\times n$ 的矩阵,可以由式(1)计算得到.其中 c 代表用户停止继续访问网页的概率,$\boldsymbol{E}$ 是单位矩阵,矩阵 $\boldsymbol{Q}$ 的元素 q_{ij} 满足式(2).则由全体网页的 PageRank 值组成的向量 $\boldsymbol{\pi}$(简称 PageRank 向量)可以根据式(3)计算得到.其中 $\underline{\boldsymbol{1}}$ 代表元素全是 1 的列向量.

$$\boldsymbol{P}=(1-c)\boldsymbol{Q}+(c/n)\boldsymbol{E}; \tag{1}$$

$$q_{ij}=\begin{cases}\dfrac{1}{k}, & k \text{ 是 } i \text{ 的出边总数},k>0\\ & \text{且 } i \text{ 有出边链接到 } j,\\ 0, & \text{其他;}\end{cases} \tag{2}$$

$$\boldsymbol{\pi P}=\boldsymbol{\pi},\ \boldsymbol{\pi}\underline{\boldsymbol{1}}=1. \tag{3}$$

当前 PageRank 算法主要可以分成 2 类:线性代数方法和蒙特卡罗方法.前者利用线性代数方法根据式(3)求解 $\boldsymbol{\pi}$,Power Iteration[5]、Gauss-Southwell[14]、基于 LU/QR 分解的 PageRank 算法[15]等是典型的线性代数计算方法.它们处理大规模矩阵效率高,且能够根据需要权衡结果精度和计算开销.蒙特卡罗方法[12,16-17]的基本思想是利用随机游走模拟大量用户随机访问网页的行为,并通过统计各节点的被访问次数估算 PageRank 的近似值.典型的蒙特卡罗方法[12]包括循环起点的全路径

算法、随机起点的蒙特卡罗算法等. 它们具有易于并行化的特点,且能够以较小的计算开销尽快地得到满足精度要求的近似结果.

相关的研究结论已证明[12]相较于其他蒙特卡罗算法,循环起点的全路径蒙特卡罗算法具有更高的计算效率,因此在已有 PageRank 算法优化研究[18]中常被作为算法改进的基础. 该算法的计算方法更适于引入增量计算的思想. 本文工作也将循环起点的全路径算法作为蒙特卡罗算法的代表,简称为基准蒙特卡罗算法,用 BasicPR 表示. BasicPR 计算 PageRank 的方法是由每个节点为起点启动 R 个随机游走过程,随机游走中的每一步以 c 的概率停止或走到等概率地随机选中的某个邻居节点,并以此选中的节点为起点继续这个随机游走过程. 对于有 n 个网页的情形,共进行 $n\times R$ 次随机游走,当所有随机游走过程都停止时,统计任意节点 u 的访问次数 $VT(u)$,用 $VT(u)$除以所有节点的访问次数的总和,就得到节点 u 的 PageRank 的近似结果. 通过增加 R,BasicPR 的结果可以足够接近 PageRank 的真实值. IncPR 使用了和 BasicPR 相同的随机游走方法并在其基础上引入了增量计算思想加以改进.

1.2 增量 PageRank 算法相关研究

为保证频繁变化的网络上获得满足时效性要求的 PageRank 值,通常需要反复进行 PageRank 计算,从而造成对计算资源的巨大消耗. 为解决这一问题,一些改进的 PageRank 算法被提出. 这些算法大致可以被归纳为 4 类:

1) 利用已有结果加速迭代收敛

Kamvar 等人[19]提出了一种基于 Aitken Δ^2 的推断方法加快收敛的算法,该算法的收敛效率受变化数据位置影响,效率的提升并不明显[18]. Ohsaka 等人[14]提出了一种通过重用已有结果的 Guass-Southwell 方法,该方法能够提升算法的收敛速度,但该算法并没有讨论节点变化的情况的处理方法.

2) 控制及利用变化的数据的影响范围

此类算法[20-23]通常将整个图按照某种策略划分成若干相互连通的子图,当某个子图内发生变化时,算法尽量将计算操作控制在子图内部,对其他子图不产生影响或影响较小,从而达到降低计算量的目的. 它们的局限性在于算法的表现受图结构的影响较大,且维护有效的图划分也比较困难.

3) 基于蒙特卡罗方法的增量改进

与线性代数方法相比,蒙特卡罗方法显然更易于按照增量计算的思想进行改进[12]. Bahmani 等人提出算法[18]保存之前的随机游走路径,当数据发生变化时,根据图的变化情况对随机游走路径进行部分更新. Bahmani 的算法假设图中的边随机地逐条到达,每当一条边到达即进行计算,当累计增加 n 条边时,处理包含 $|V|$ 个节点的图数据,该算法的时间复杂度为 $O(|V|\ln n/c^2)$. Lofgren 等人[24]在 Bahmani 等人工作[18]的基础上,对算法复杂度的分析进行了补充. 该算法是一种高效的增量 PagaRank 算法,能有效应对节点和边发生变化的情况,但其保存随机游走路径的历史数据的做法也引入了较大的存储开销.

4) 其他方法

Tong 等人[25]提出了一种适合于二分图的更新 PageRank 值的方法,该方法在 $|V|\times l$ 的图上能够获得 $O(l^2)$ 的时间复杂度,然而互联网和社交网络中的 $l=|V|$,当 n 条边到达时,更新的代价为 $O(n|V|^2)$,并不适合扩展到大规模图数据中来. Mcsherry 等人[26]将多种启发算法应用到迭代算法中,以增量的更新操作计算流数据上的 PageRank 值,但该算法在控制计算量的情况下不能保证结果精度.

综上,已有的增量 PageRank 算法,都需要消耗额外的存储空间. 本文提出的 IncPR 并不需要额外的存储空间,且相较于已有的高效增量 PageRank 算法,IncPR 的时间复杂度更低,在计算效率和存储开销 2 个方面优势明显.

2 增量 PageRank 计算模型

为降低频繁变化的海量数据集上反复计算 PageRank 的资源消耗,节约计算成本,本文提出了一种增量 PageRank 算法——IncPR. 本节首先以一个示例介绍基于增量计算思想的 PageRank 计算方法,接着给出 IncPR 的计算模型的形式化概述.

2.1 增量 PageRank 计算示例

增量 PageRank 计算将每一次数据更新作为一个新时刻. 假设时刻 t 图 G 形如图 1(a)所示,其中实线表示时刻 $t-1$ 的图数据,虚线表示的边 $e_{3,5}$ 是在时刻 t 新增加的. 图 1(b)表示时刻 $t-1$ 使用 BasicPR 得到的随机游走路径的集合.

在使用 BasicPR 计算时刻 t 的图数据时,节点 5 在随机游走中被访问的机会较时刻 $t-1$ 增大. 如果以图 1(b)所示的随机游走路径当作时刻 t 的初始路径,那么仅需要更改部分路径(每次访问节点 3 后

的随机游走路径)即可以得到符合时刻 t 各节点的被访问概率的随机游走路径.即每当遇到随机游走路径中出现节点 3,即按照时刻 t 的图,为节点 3 重新选择下一步的节点,并更新之后的随机游走路径.这样即可得到如图 1(c)所示的随机游走路径.尽管这个集合是由图 1(b)变化来的,但该集合中各节点被访问的概率和在时刻 t 的图上重新产生的随机游走路径集合相等,其概率都是由图的拓扑特征和随机游走的停止概率参数共同决定的.

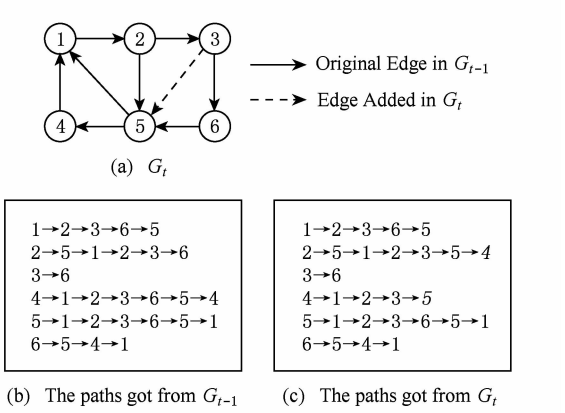

Fig. 1 An example of computing PageRank incrementally.

图 1 增量 PageRank 计算示例

基于这种考虑,就可以将增量计算的思想引入 PageRank 的计算过程.以时刻 $t-1$ 的 BasicPR 得到的随机游走路径为基础,在发生变化的边(或节点)所影响的路径上采用相同的随机游走过程更新随机游走路径,并重新计算 PageRank 得到新结果.这样的增量计算方式,避免了重新从头生成全图的随机游走路径,降低了计算量.这也是已有的基于蒙特卡罗方法的增量 PageRank 算法[12,18,24]的基本计算过程.此类算法能够降低 PageRank 的计算量,但需要较大的存储开销来保存随机游走路径.

而从节点的访问概率的角度观察,对时刻 t 的图进行计算时,与时刻 $t-1$ 相比,每次访问节点 3 后,节点 5 被访问的概率增加,节点 6 被访问的概率减少,而且节点 5 被访问的概率的增加量和节点 6 减少的量基本相当.这一特点即是 IncPR 的设计基础之一.

2.2 增量 PageRank 计算模型概述

IncPR 是一种基于增量思想的 PageRank 算法,易于并行化实现.与已有的基于蒙特卡罗算法的增量 PageRank 算法不同,IncPR 并不需要保存任何随机游走路径的历史信息.它利用之前一个时刻已经得到的 PageRank 向量,逆推得到蒙特卡罗模拟的节点访问次数统计,结合图数据的变化情况,增量地更新得到新图上的节点访问次数统计,进而更新 PageRank 的计算结果.

IncPR 的计算过程借助于这样的事实,即蒙特卡罗算法中节点的访问次数的期望具有稳定性[11],来保证其正确性(相关证明见 4.1 节).IncPR 的计算过程本质上与 2.1 节中的增量计算过程类似,即以受增量影响的任意节点 v 为起点,生成若干新路径片段,并用新路径片段替换旧路径片段的过程.

IncPR 能有效地处理图中节点和边的增加与删除的情况,且由于能较大程度地利用之前的结果,降低重复计算,因此该算法能够有效地减少图数据变更后重计算 PageRank 值的计算量.

IncPR 在变化的图数据上进行增量的 PageRank 计算的过程,可以形式化地描述为以下的增量计算工作模型,如图 2 所示.对于图 $G(V,E)$,令时刻 t 图的快照为 $G_t(V_t,E_t)$.图的变化部分 $\Delta G_t=diff(G_{t-1},G_t)=\{\Delta V_t,\Delta E_t\}$.相应地,从 G_t 得到的 PageRank 结果记作 $\boldsymbol{PR}_t$.IncPR 假设 ΔG_t 已知,$\boldsymbol{PR}_{t-1}$ 已知,并通过预处理操作 $pre(\boldsymbol{PR}_{t-1})$ 得到 BasicPR 处理 G_{t-1} 时得到的所有节点的访问次数的集合 $\boldsymbol{VT}_{t-1}$.

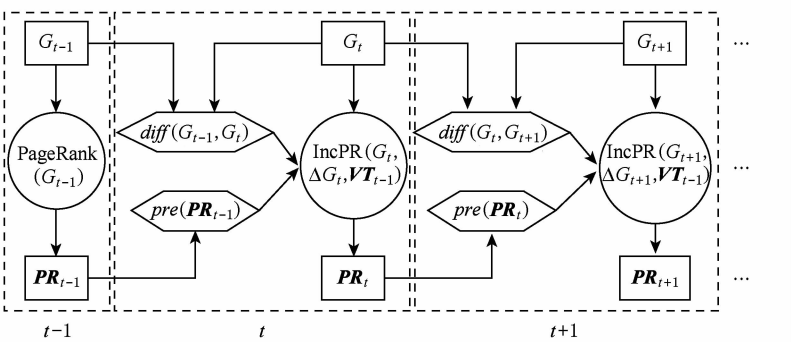

Fig. 2 Overview of incremental computation of PageRank.

图 2 增量 PageRank 计算模型

IncPR 算法以 G_t, ΔG_t 和 $\boldsymbol{VT}_{t-1}$ 为输入，在 $\boldsymbol{VT}_{t-1}$ 的基础上，结合 G_t, ΔG_t 采用改进的蒙特卡罗方法增量地更新 ΔG_t 对于 PageRank 新值的影响，最终得到新的 PageRank 计算结果 PR_t. 一般使用 X_t 代表符号 X 在时刻 t 对应的对象集合，而使用 $|X|$ 代表集合 X 的元素的个数.

$$\boldsymbol{VT}_t = SVT_t \times \boldsymbol{PR}_t. \tag{4}$$

预处理可以按照式(4)的方式求得 $\boldsymbol{VT}_{t-1}$. 其中，SVT_t 是时刻 t 的蒙特卡罗模拟中所有节点访问次数的总和，$SVT_t = |V_t|R/c$. 值得注意的是，式(4)具有普遍性，即使 $\boldsymbol{PR}_{t-1}$ 并不是通过蒙特卡罗方法得到，也可以通过式(4)得到 $\boldsymbol{VT}_{t-1}$. 也就是说 IncPR 可以利用任意 PageRank 算法得到的结果.

3 IncPR：增量 PageRank 算法

IncPR 通过重用当前已有的计算结果初始化节点的访问次数，根据图数据的变化对最终结果进行计算. 对于图数据变化的不同情况，采用不同的计算策略. 本文将图的变化分为 2 种情况：1)边的变化，仅有边的增加和删除；2)点的变化，节点和边均有增加和删除. 本节将分别介绍对这2种情况的处理方法.

3.1 IncPR 处理边的变化

IncPR 处理边的变化时采取 2 个主要步骤：

1) 对于每条变化的边，按一定策略更新某些节点的访问次数；

2) 访问次数被更新的节点，按照访问次数的更新量进行若干次随机游走，将新生成的随机游走路径上的节点的访问次数(依据一定的原则)增或减 1.

图 3 展示了一个节点上边的变化的 3 种典型情况(与边的变化无关的节点和边被省略)，图 3(a)表示时刻 $t-1$ 的图数据 G_{t-1}. 图 3(b)(c)(d)表示时刻 t 的图数据 G_t，分别描绘了增加一条边、删除一条边、同时增加删除多条边的示例.

增加一条边的情况，如图 3(a)(b)所示，$\Delta G_t^1 = \{e_{u,v}^+\}$. 2 次计算相比，当节点 u 被访问时，其子节点被访问的概率会有所变化. 在时刻 $t-1$，每当节点 u 被访问时，节点 w, s 分别以 $(1-c)/2$ 的概率被访问；在时刻 t，每当节点 u 被访问时，节点 w, s, v 分别以 $(1-c)/3$ 的概率被访问. 边 $e_{u,v}$ 的增加使得节点 v 的访问次数的期望增加了 $(1-c)VT(u)_{t-1}/|S(u)_t|$，节点 w, s 的访问次数的期望减少了 $(1-c)\times VT(u)_{t-1}(1/|S(u)_{t-1}|-1/|S(u)_t|)$，其中 $S(u)$ 表示时刻 t 节点 u 出边指向的节点的集合.

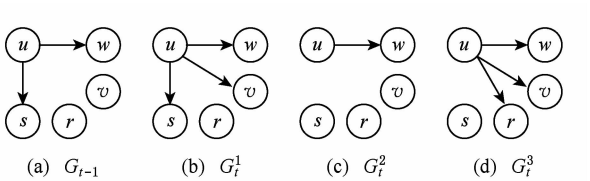

Fig. 3 Three cases of changing edges.

图 3 边变化的 3 种情况

为了利用时刻 $t-1$ 的访问次数得到时刻 t 的结果，节点 v 的访问次数需要加上期望的变化量，节点 w, s 的访问次数平均分摊 $VT(v)_t$ 的增加量，作为各自访问次数的减少量. 当 $VT(v)_t$, $VT(w)_t$ 和 $VT(s)_t$ 都更新完毕，则按照步骤 2 操作，s, v 和 w 分别进行与各自访问次数更新数量相等的随机游走，从 v 发出的新随机游走路径上的节点的访问次数增 1，表示新路径的添加；从 s 和 w 发出的新随机游走路径上的节点将访问次数减 1，表示旧路径的删除. 上述操作的结果等价于经过节点 u 的部分路径被新路径替换. 总体上看，上述操作相当于由于边的增加，以 s, v 和 w 三节点为起点的随机游走路径中，从 u 获得的访问次数的重分配，重分配的结果与在图 G_t 上重新进行随机游走分配节点 u 发出访问的情况相当.

边删除情况与边增加情况的处理办法类似. 区别仅在于访问次数的增减数量和操作顺序. 如图 3(a)(c)所示，当边 $e_{u,s}$ 被删除时，节点 s 的访问次数 $VT(s)_t$ 减少，减少量的期望等于 $(1-c)\times VT(u)_{t-1}/|S(u)_{t-1}|$，相应的节点 w 的访问次数 $VT(w)_t$ 增加，增加量的期望等于 $VT(s)_t$ 的减少量. 之后，分别以 w, s 为起点进行对应数量的随机游走，来更新节点 w, s 的变化对其子节点的影响. 以 w, s 为起点的随机游走经过的节点分别进行加 1 和减 1 的操作表示新路径的生成和旧路径被替换.

结合边的增加和删除的情况，可以得到更一般化的处理方法. 令 $e_{u,x}$ 表示变化(增加或删除)的边，M 代表 $VT(x)_t$ 变化量的大小，对于 u 的其他邻居 y，令 N 代表 $VT(y)_t$ 变化量的大小，如式(5)(6)所示：

$$M = \frac{(1-c)VT(u)_{t-1}}{|S(u)_{t-1} \cup S(u)_t|}, \tag{5}$$

$$N = \begin{cases} M/|S(u)_{t-1}|, & e_{u,x} \in \Delta E^+, \\ M/|S(u)_t|, & e_{u,x} \in \Delta E^-. \end{cases} \tag{6}$$

图 3(d)反映了在时刻 t 分别增加和删除多条边的情况. 由蒙特卡罗方法的计算特点可知，多条边分别增加和删除的情况，可以视作多次发生增加单条

边和删除单条边的情况,按照上述处理单条边增加和删除的办法依次处理即可.

算法 1 使用伪代码描述了 IncPR 针对图中边的变化的处理逻辑.

算法 1. ProcessIncEdges(G_t,$diff(G_{t-1},G_t)$,VT_{t-1},c).

输入:$G_t=\{E_t,V_t\}$表示时刻 t 的图数据、$diff(G_{t-1},G_t)=\{diff(E)^+,diff(E)^-\}$代表时刻 $t-1$ 到时刻 t 图数据的变化量(边的增加和删除)、VT_{t-1} 代表时刻 $t-1$ 的节点访问次数的集合、c 是随机游走的停止概率;

输出:VT_t 为图 G_t 的节点访问次数的集合.

① 初始化 $VT_t=VT_{t-1}$;
/* 过程 1 */
② List $SRW\_Array\langle node,count,tag\rangle$;
③ for $e_{u,v}\in diff(G_{t-1},G_t)$ do
④ 由式(5)(6)计算 M,N;
⑤ if $e_{u,v}\in diff(E)^+$ do
⑥ $SRW\_Array.add(v,M,+)$;
⑦ for $w\in S(u)_{t-1}$ do
⑧ $SRW\_Array.add(w,N,-)$;
⑨ end for
⑩ else
⑪ $SRW\_Array.add(v,M,-)$;
⑫ for $w\in S(u)_t$ do
⑬ $SRW\_Array.add(w,N,+)$;
⑭ end for
⑮ end if
⑯ end for
/* 过程 2 */
⑰ for 任意$\langle x,count,tag\rangle$ in SRW_Array do
⑱ while $count>0$ do
⑲ 以x 为起点进行停止概率为c 的随机游走,得到随机游走路径 L;
⑳ if $tag==$"+" do
㉑ 对任意节点 y in L,$VT(y)_t$++;
㉒ else
㉓ 对任意节点 y in L,$VT(y)_t$--;
㉔ end if
㉕ $count$--;
㉖ end while
㉗ end for

3.2 IncPR 处理点的变化

节点的增加通常伴随边的增加,可分为新增节点有入边和无入边 2 种情况讨论. 如果新增加的节点没有入边,则只要按照与时刻 $t-1$ 相同的随机游走方法补上 R 次以新节点为起点的随机游走,再更新节点的 PageRank 值即得到最终结果. 但如果新增节点有入边,则需要在上述处理的基础上按照 3.1 节的方法处理. 如图 4 所示,图 4(a)(b)分别表示时刻 $t-1$ 和时刻 t 的图数据,这种情况可以视作 2 个过程的叠加:先增加一个节点和节点的出边,得到如图 4(c)所示的过渡图数据和对应的 PageRank 结果;再增加节点的入边.

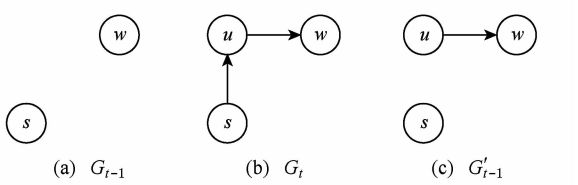

Fig. 4 An example of disassembly of adding a node.

图 4 节点增加等价于节点和边分别增加

类似地,节点的删除则可以看作 2 个过程:节点的删除和与节点相连边的删除. 而稍有不同的是,对于删除节点的情况,只需对变化的边按照 3.1 节中的方法处理,而不需要再重复地减去 R 次以被删除节点为起点的随机游走对其他节点的影响,这个操作的影响在处理被删除的边时已被处理.

综上,本文提出的 IncPR 可以概括为 2 个步骤:1)处理新增节点和节点的出边;2)按照 3.1 的方法处理变化的边. 算法 2 为 IncPR 的伪代码描述.

算法 2. IncPR(G_t,$diff(G_{t-1},G_t)$,VT_{t-1},c,R).

输入:$G_t=\{E_t,V_t\}$表示时刻 t 的图数据、$diff(G_{t-1},G_t)=\{diff(E)^+,diff(E)^-,diff(V)^+,diff(V)^-\}$为时刻 $t-1$ 到时刻 t 图数据的变化量(边和节点的增加和删除)、VT_{t-1} 为时刻 $t-1$ 得到的被访问次数的集合、c 是随机游走的停止概率、R 是各节点开始的随机游走次数;

输出:PR_t 为图 G_t 的 PageRank 向量.

① 初始化 $VT_t=VT_{t-1}$; /* 新增节点的访问次数为 0 */
/* 过程 1 */
② for $u\in diff(V)^+$ do
③ $count=R$;
④ while $count>0$ do
⑤ 以u 为起点进行停止概率为c 的随机游走,得到随机游走路径 L;
⑥ 对任意节点 y in L,$VT(y)_t$++;
⑦ $count$--;

⑧ end while
⑨ end for
⑩ $G'_{t-1}=\{E_{t-1},V_t\}$
/*过程 2*/
⑪ ProcessIncEdges(G_t,$diff(G'_{t-1},G_t)$,VT_t,c);
⑫ $SVT_t=|V_t|R/c$
⑬ for $u\in V_t$ do
⑭ $PR(u)_t=VT(u)_t/SVT_t$;
⑮ end for

4 算法正确性及时间复杂度分析

4.1 算法的正确性证明

IncPR 是一种增量的蒙特卡罗模拟算法,基于模拟的 PageRank 算法得到的结果是近似值.类似于文献[11,18]中对 BasicPR 正确性的证明,本文证明了 IncPR 与 BasicPR 计算得的节点的访问次数的期望相等且 IncPR 计算得到节点的访问次数与真实的访问次数足够接近,这样即证明 IncPR 是正确的.

定理 1. 对于任意节点 v,在图 G_t 上使用 IncPR 得到的访问次数的期望 $E[VT(v)_{\text{IncPR}}]$,与使用 BasicPR 计算得到的访问次数的期望 $E[VT(v)_{\text{BasicPR}}]$ 相等.

证明. IncPR 在进行计算时,利用图 G_{t-1} 的访问次数初始化 G_t 中节点的访问次数,初始化的结果可以看作是从 $|V_{t-1}|$ 个节点出发每个节点进行 R 次随机游走得到的 $|V_{t-1}|R$ 条路径.对于每一个变化的边 $e_{u,v}$,IncPR 会从节点 u 的子路径中选出 M 条旧的子路径,并生成 M 条新的子路径替换旧路径.此操作并没有改变总路径的起点和数量;对于每个变化的节点,IncPR 沿用 BasicPR 从每个节点进行 R 次随机游走.因此,IncPR 处理完 G_t 后,可以看作得到与 BasicPR 起点相同且对应起点数目相同的路径.

使用 BasicPR 计算图 G_t 的 PageRank 值时,图中任意节点 v 的访问次数的期望为

$$E[VT(v)_{\text{BasicPR}}]=\sum_{u\in V_t}\alpha_u P_{u,v}=R\sum_{u\in V_t}P_{u,v},\quad(7)$$

其中,$P_{u,v}$ 表示以节点 u 为起点的随机游走路径经过节点 v 的概率,α_u 表示以节点 u 为起点的随机游走次数,由于 2 种算法每个节点都进行了 R 次随机游走,故任意节点 u 的 $\alpha_u=R$.从节点 u 出发的随机游走路径经过节点 v 的概率 $P_{u,v}$ 是由每一条从节点 u 到节点 v 的路径的概率累加得到的,如式(8)所示:

$$P_{u,v}=\sum_{w\in S_t(u)}q_{u,w}q_{w,i}\cdots q_{j,v},\quad(8)$$

其中,q_{ij}($i=u,w,\cdots,y$;$j=w,x,\cdots,v$)如式(2)所示,表示节点 u 到节点 w 的跳转概率,节点 u 选取某一条路径访问节点 v 的概率为该路径上从 u 到 v 经过的节点间的跳转概率的乘积.BasicPR 在进行随机游走时,任意节点间的跳转概率只与图结构有关,若节点 u,w 之间存在连边,则 $q_{u,w}=(1-c)/|S(u)|$;IncPR 在进行随机游走时,对于出边发生变化的节点 u,该节点的访问次数按照期望值在子节点之间均匀分配,节点 u 到任意子节点的跳转概率为 $(1-c)/|S(u)|$,这些节点与子节点间的跳转概率与 BasicPR 中的相同;其他节点间的跳转概率按照 $q_{u,w}$ 进行处理,与 BasicPR 相等.因此,IncPR 在得到 $|V_t|R$ 条路径时,任意节点间的跳转概率均与 BasicPR 相同.2 种算法在计算图 G_t 的 PageRank 值时,任意节点间的 $P_{u,v}$ 相等.

综上,IncPR 算法与 BasicPR 在计算相同图数据的 PageRank 值时,任意节点的访问次数的期望相等,定理 1 得证. 证毕.

定理 2. 对于任意节点 v,在图 G_t 上使用 IncPR 得到的节点 v 的 PageRank 值与节点 v 的 PageRank 的真实值非常接近,即有式(9)成立:

$$P[|\tilde{\pi}_v-\pi_v|\geqslant\delta\pi_v]\leqslant e^{-nR\pi_v\delta'},\quad(9)$$

其中,$\tilde{\pi}_v$ 和 π_v 分别表示 IncPR 得到的节点 v 的 PageRank 值和节点 v 的 PageRank 的真实值.类似的定理在文献[11,18]中已得到过证明,但为保证本文论述的完整性,仍在此引述相关的证明过程.

证明. 先讨论 $R=1$ 的情况($R>1$ 的情况同理可证).假设随机变量 X_u 是由 u 出发的随机游走路径上节点 v 的访问次数的 c 倍.Y_u 是这条随机游走路径的长度.令 $W_u=cY_u$,$x_u=E[X_u]$.由于对于不同的节点 u 随机变量 X_u 相互独立,则有:

$$\tilde{\pi}_v=\sum_u X_u/n,\qquad\pi_v=\sum_u x_u/n,\quad(10)$$

显然有:$0\leqslant X_u\leqslant W_u$,$E[W_u]=1$.

则由期望的定义可知:

$$\frac{E[e^{tX_u}]-1}{E[e^{tW_u}]-1}\leqslant\frac{E[X_u]}{E[W_u]},\quad(11)$$

进而得到:

$$E[e^{tX_u}]\leqslant x_uE[e^{tW_u}]+1-x_u=x_u(E[e^{tW_u}]-1)+1.\quad(12)$$

由于对任意的 y 有 $1+y\leqslant e^{y}$,因此可得到:

$$E[e^{tX_u}]\leqslant e^{-x_u(1-E[e^{tW_u}])}. \tag{13}$$

由马尔可夫不等式,可以得到:

$$P[\tilde{\pi}_v\geqslant(1+\delta)\pi_v]\leqslant E[e^{tn\tilde{\pi}_v}]/e^{tn(1+\delta)\pi_v}=$$
$$E[e^{t\sum_u X_u}]/e^{tn(1+\delta)\pi_v}=\prod_u E[e^{tX_u}]/e^{tn(1+\delta)\pi_v}\leqslant$$
$$\prod_u e^{-x_u(1-E[e^{tW_u}])}/e^{tn(1+\delta)\pi_v}=$$
$$e^{-(\sum_u x_u(1-E[e^{tW_u}]))}/e^{tn(1+\delta)\pi_v}=$$
$$e^{-n\pi_v(1-E[e^{tW_u}])}/e^{tn(1+\delta)\pi_v}=$$
$$e^{-n\pi_v(1+t(1+\delta)-E[e^{tW_u}])}\leqslant e^{-n\pi_v\delta'}. \tag{14}$$

类似地可以证明:

$$P[\tilde{\pi}_v\leqslant(1-\delta)\pi_v]\leqslant e^{-n\pi_v\delta'}, \tag{15}$$

则定理 2 得证. 证毕.

综合定理 1,2 可知,IncPR 得到的计算结果能够确保与 BasicPR 得到的结果同等正确. BasicPR 计算 PageRank 的结果正确性在文献[12]中已被证明,因此 IncPR 也具有与 BasicPR 同等的正确性.

4.2 算法的时间复杂度分析

本节通过分析 IncPR 的处理过程,推导出其时间复杂度. 使用 IncPR 计算 PageRank 向量时,对于每一个增加的节点,算法都会以该节点为起点进行 R 次随机游走;对于每一个变化的边 $e_{u,v}$,都会以 v 为起点进行 M 次随机游走,作为新的子路径;同时,会以 u 的其他子节点为起点进行 M 次随机游走,作为待被替换子路径. 一条变化的边需要进行 $2M$ 次随机游走,以更新结果.

因此,假设 G_t 在 G_{t-1} 的基础上增加了 m 个节点且增加和删除了共计 n 条边,则增量算法需要进行随机游走的总次数 $TotalRW$ 满足式(16),其中 M 按式(5)求得.

$$TotalWR=mR+2\sum_{e_{u,x}\in\Delta E}M=$$
$$mR+\sum_{e_{u,x}\in\Delta E}\frac{2(1-c)VT(u)_{t-1}}{|S(u)_{t-1}\cup S(u)_t|}. \tag{16}$$

令 $TotalComp$ 表示 IncPR 的计算量大小,则有式(17),其中 $E[X]$ 代表随机游走路径长度的期望值.

$$TotalComp=TotalRW\times E[X]. \tag{17}$$

由于图中所有节点至少包含一个出边(悬挂点被看作包含 $|V_t|$ 个出边),发生变化边的起点至少增加或删除了一条边. 因此对于任意变化的边 $e_{u,v}$ 的起点 u,有:

$$|S(u)_{t-1}\cup S(u)_t|\geqslant 2, \tag{18}$$

因此,可进一步推导出:

$$TotalComp\leqslant[mR+\sum_{e_{u,v}\in\Delta E}VT(u)]E[X], \tag{19}$$

$$\sum_{e_{u,v}\in\Delta E}VT(u)\approx\sum_{e_{u,v}\in\Delta E}E[VT]=E[VT]\times n. \tag{20}$$

令 $E[VT]$ 表示节点访问次数的均值. 节点访问次数与节点总数的乘积为总的访问次数,总的访问次数等于随机游走路径总长度,故 $E[VT]=RE[X]$. 由于随机游走路径长度服从参数为 c 的几何分布,故 $E[X]=1/c$,$E[VT]=R/c$. 因此有:

$$TotalComp\leqslant Rm/c+Rn/c^2. \tag{21}$$

当图 G 发生变化的增量包含 m 个节点和 n 条边时,IncPR 算法根据图 G 的中间数据更新结果的时间复杂度为 $O((cm+n)R/c^2)$.

5 性能测试与分析

5.1 实验设置

本文分别以 Bahmani 提出的增量 PageRank 算法[19]和 BasicPR 作为比较基准,对 IncPR 的结果正确性和计算量大小进行了实验. 实验在基于 BSP 计算模型的 Hama 分布式计算框架[27]上实现了 IncPR 和上述 2 种对比算法. Hama 的体系结构如图 5 所示. 实验使用的 Hama 集群由 6 台同构节点组成,每个节点配备 intel-i7 2600 CPU、8 GB 内存、2 TB 磁盘,并通过千兆以太网连接. 各节点安装 Ubuntu14.04 操作系统,主要的计算工具版本分别为 Hama-0.6.4、zookeeper-3.4.6 和 Hadoop-1.2.1.

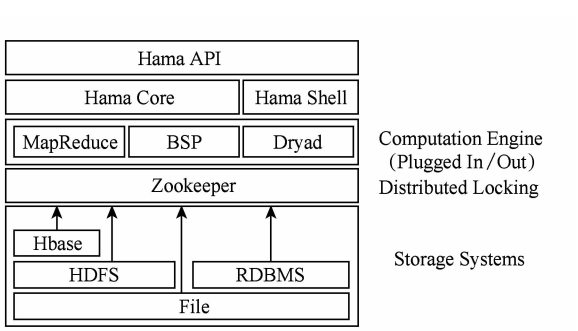

Fig. 5 Overview architecture of Hama.

图 5 Hama 分布式计算框架体系结构

实验使用了多组具有不同应用背景的真实数据集,对典型的数据集变化情况分别进行了多组实验,实验结果验证了 IncPR 的正确性和计算效率. 权衡 PageRank 结果的精度和实验运行时间,实验选取

参数$R=20$,$c=0.15$,默认情况下,初始的 PageRank 结果由该参数下的 BasicPR 计算得到.

5.2 评价指标

实验主要从算法的结果精度和计算量 2 个方面考虑,选取了 2 个评价指标,以测试分析 IncPR 的正确性和计算效率.

1) 正确性指标

类似于已有的相关研究工作[10,14],本文定义算法正确性的评价指标 $Err(Alg)$ 如式(22). 其中 $PR(u)$代表由算法 Alg 得到的 PageRank 的计算结果,$\pi(u)$代表 PageRank 的真实值. 显然地,$Err(Alg)$的大小代表了算法 Alg 的结果的相对精度. 对于相同的图数据,$Err(Alg)$越小意味着算法得到的 PageRank 的计算结果的精度越高,即越接近真实的 PageRank 值. 本文实验中 PageRank 的真实值由 Power Iteration 算法计算获得,参数 $\varepsilon=0.0005$.

$$Err(Alg)=\frac{\sum_{u\in V}|PR(u)-\pi(u)|}{\sum_{u\in V}\pi(u)}. \quad (22)$$

2) 计算量指标

本文定义 $Cost(Alg)$作为算法计算效率的评价指标. 基于蒙特卡罗的 PageRank 算法通过进行大量的随机游走来计算 PageRank 值,构造随机游走路径是此类算法的主要开销. 由于随机游走中每一步的开销相同,算法的计算量可以由产生的随机游走的总步数表示. 本文实验基于 Hama 框架实现,PageRank 算法通过传递消息实现随机游走,通过统计消息通信的总数即得到随机游走的总步数. $Cost(Alg)$的值即等于算法在 Hama 框架上的发送的消息的总数.

5.3 数据集及预处理

实验使用了多组不同的真实图数据集[28],它们覆盖了从 P2P 网络到 Web 互联网和社交网络等多个不同应用领域. 数据集的主要特征参数详见表 1 所示. 类似于文献[10]中的做法,在不影响实验结果的准确性和算法计算量的前提下,实验对图数据进行了一定的预处理,将图中所有的单向边都改为双向边.

图数据的变化包括点和边的添加和删除共 4 种情况. 由于增量算法在处理增加删除边的 2 种操作类似(区别仅是在进行随机游走时经过节点的操作),因此实验仅测试添加点和边情况下算法的正确性和计算效率. 仅增加边的实验中,先从原图中随机选择最多 10%的边删除,将得到的图作为初始数据 G,再从被删除边中,按固定比例(0.01%,0.05%,0.1%,0.5%,1%,5%,10%)随机添加到图 G 中作为变化后的图数据. 增加点的实验,以原图为初始数据 G,以 G 中的节点数为基准,生成固定比例(0.01%,0.05%,0.1%,0.5%,1%,5%,10%)的新节点并添加到 G 中,随机地为新节点选择一条入边和出边.

Table 1 Main Parameters of Data Sets

表 1 数据集的主要特征参数

Data Set	Background	$10^{-3}\times$ Nodes	$10^{-3}\times$ Edges	$10^{-3}\times$ Dangling Nodes
P2P-Gnutella31	P2P	63	148	46
amazon0312	Online Business	401	3 200	12
web-NotreDame	Web	326	1 497	187
web-BerkStan	Web	685	7 600	4.7
higgs-twitter	Social Network	457	14 855	0.03
wiki-talk	Wikipedia	2 394	5 021	2 200
wiki-vote	Wikipedia	7.1	104	1
email-Enron	Communication Network	37	184	—

5.4 结果准确性

本节通过比较 IncPR 与 Bahmani 的增量算法和 IncPR 与 BasicPR 计算 PageRank 值的相对误差的大小,来验证 IncPR 的正确性. 在比较 IncPR 与 Bahmani 的增量算法的正确性时,分别增加不同数量的边比较 2 种算法的误差,实验表明 2 种算法具有相同量级的精度水平. 在比较 IncPR 与 BasicPR 的正确性时,考虑在真实的数据集上,仅增加边和增加点和边 2 种情况下算法的正确性. 此外,通过多组不同的实验,验证了不同的图数据变化量、不同的初始结果精度对 IncPR 结果精度的影响.

如图 6 所示,IncPR 在结果精度上表现良好. 当增量规模较小时(小于 1%),IncPR 和 BasicPR 的结果的误差接近,随着增加节点的比例的增大,IncPR 与 BasicPR 的误差的比在逐渐减小,这意味着 IncPR

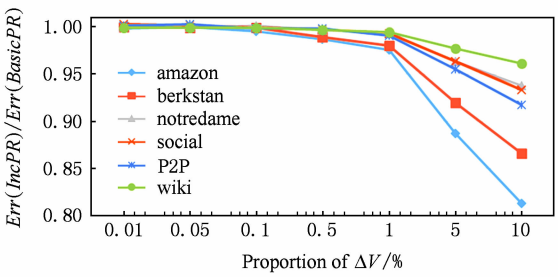

Fig. 6 Comparison of $Err(Alg)$ with different proportion of ΔV.

图 6 增加不同比例的节点的误差比较

在增加的点较多的情况下(5%和 10%)得到比 BasicPR 更高的结果精度.最优情况下,IncPR 的误差比 BasicPR 减小接近 20%.

图 7 反映了增加不同比例的边(占原图的 0.01%到 10%)的情况下,IncPR 与 BasicPR 得到的结果误差的比值.边增加的情况下 2 种算法的误差总体接近,仅有个别图得到了最好情况下接近 10%的精度提升.另一组增加不同数量的边的实验结果(如图 8 所示),也同样验证了 IncPR 得到与 BasicPR 的结果精度在相同量级的结论.

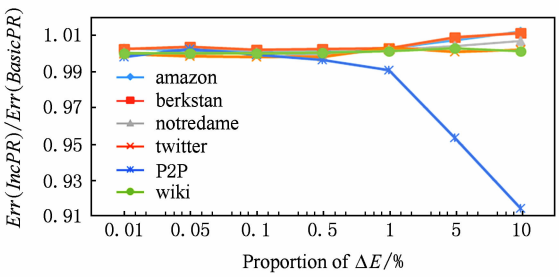

Fig. 7 Comparison of $Err(Alg)$ with different proportion of ΔE.

图 7 增加不同比例的边的误差比较

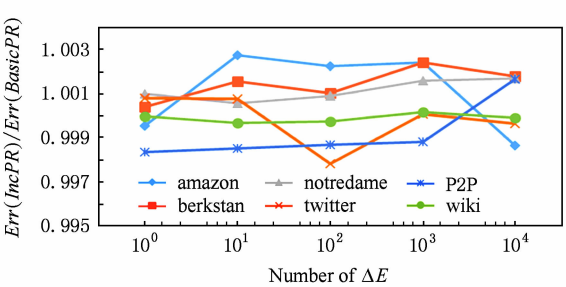

Fig. 8 Comparison of $Err(Alg)$ with different number of ΔE.

图 8 增加不同数量的边的误差比较

对于 IncPR 比 BasicPR 获得的精度提升,一种合理的解释是图的变化一定程度地影响了其拓扑结构,BasicPR 的结果精度受图的变化的影响而略有上升,而 IncPR 采用增量的更新方法,避免了由图的变化而引入更大的误差,从而得到了更好的结果精度.图 9 所反映的随着节点的增加 BasicPR 的误差上升,一定程度支持了这种解释.

此外,IncPR 的一个有价值的特点是,其可以通过重用误差较小的数据获得精度较高的计算结果.这意味着如果以较大的计算代价计算出高精度的 PageRank 结果,再使用 IncPR 处理变化后的新图,就可以利用较低的成本获得新的高精度结果,相当

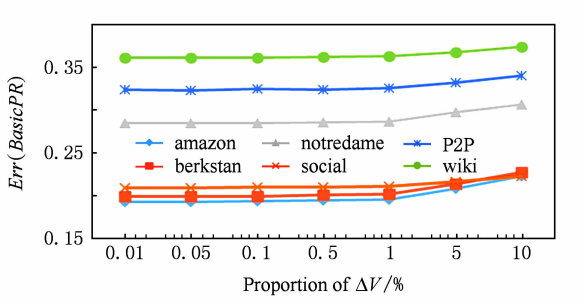

Fig. 9 $Err(BasicPR)$ with different proportion of ΔV.

图 9 增加不同比例的节点的 BasicPR 的结果误差

于将蒙特卡罗方法适于处理增量和适于并行化的特点与其他的计算代价大但结果精度高的 PageRank 算法结合了起来,可以充分发挥二者各自的优势.

图 10 和图 11 反映了当以 PI 计算得到的 PageRank 向量作为初始结果时,增加固定比例的节点和边的情况下分别得到的 IncPR 的结果误差与 BasicPR 的误差的比.显然地,当图的变化量较小时(不超过 1%),IncPR 得到的结果精度显著优于 BasicPR,IncPR 得到的结果精度接近 PI 算法,当增量较大时(5%,10%),最差情况 IncPR 的误差也不到 BasicPR 的一半.

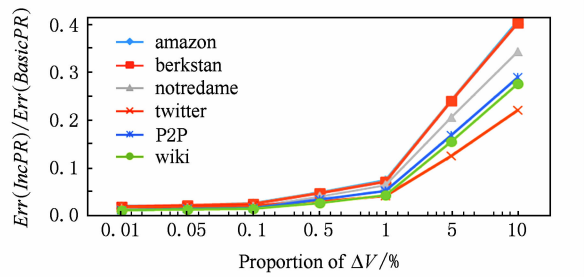

Fig. 10 Comparison of $Err(Alg)$ with different proportion of ΔV (IncPR Starts from PI).

图 10 增加不同比例的节点误差比较(以 PI 结果作为初始值)

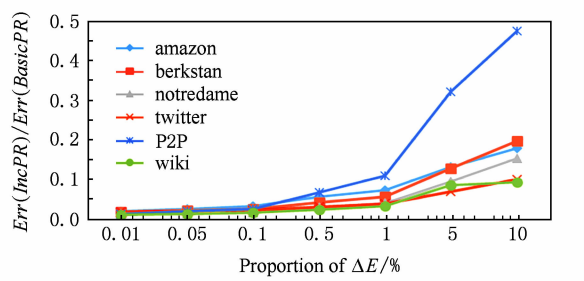

Fig. 11 Comparison of $Err(Alg)$ with different proportion of ΔE (IncPR Starts from PI).

图 11 增加不同比例的边的误差比较(以 PI 结果作为初始值)

由于篇幅所限,本文没有给出 IncPR 与 Bahmani 的增量算法的正确性比较的数据,2 种算法的结果误差值相近.

5.5 计算量对比

本文通过多组真实数据集上不同场景的实验,分别比较了 IncPR 与 Bahmani 的增量算法及 IncPR 与 BasicPR 的计算量的大小,作为评价算法计算效率的标准.

首先对 IncPR 与 Bahmani 的增量算法的计算量进行了对比实验,实验结果如表 2 所示. 实验中令增加的边数 n 分别为 50,100 和 150,由于 Bahmani 的增量算法每增加一条边就更新结果,因此,实验中统计执行 n 次 Bahmani 的增量算法累积的计算量. 实验结果均表明 IncPR 的计算量明显优于 Bahmani 的增量算法,实验结果验证了 IncPR 的时间复杂度优于 Bahmani 的增量算法. 虽然与 Bahmani 的增量算法的对比实验中使用的 n 较小,但由理论分析可知,IncPR 的计算量与 n 的大小呈线性关系,而 Bahmani 的增量算法的计算量与节点总数 $|V|$ 的 $\ln n$ 倍呈线性关系,网络数据中增量的比例占原数据比例较小,$n \ll |V|$. 因此在 n 较大的情况下应该会得到类似的结果.

Table 2 Performance Comparison of Bahmani's and IncPR

表 2 Bahmani 增量算法与 IncPR 计算量比较 K

Data Set	50 edges added		100 edges added		150 edges added	
	Bahmani	IncPR	Bahmani	IncPR	Bahmani	IncPR
P2P-Gnutella31	205	6.1	362	11.3	644	17.4
email-Enron	146	15.5	292	31.5	424	47.7
wiki-vote	211	3.0	312	5.7	515	8.9

进而,本文分别在不同的数据变化量、初始结果精度等多种条件下进行实验,比较 IncPR 与 BasicPR 计算 PageRank 值的计算量大小,得到算法计算量的统计,作为评价算法计算效率的标准.

图 12 与图 13 分别反映了增加固定比例的节点和边的情况下,IncPR 与 BasicPR 的计算量的比值. 2 组实验数据都表现出相同的规律,IncPR 的计算量明显小于 BasicPR. 当图的变化量不超过 0.01% 时,IncPR 的计算量最大仅为 BasicPR 的 0.09%,当图的变化量达到 10%时,IncPR 的计算量最大仅相当于 BasicPR 的 20%. 实验结果表明 IncPR 能有效地降低 PageRank 计算的计算量,提高计算效率.

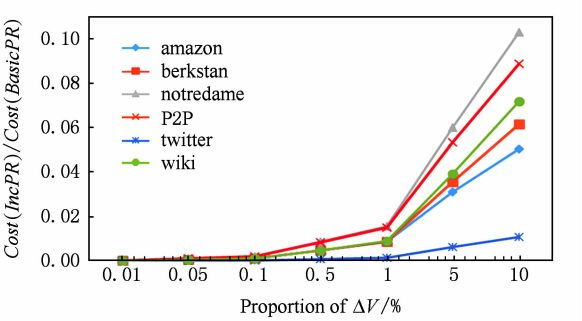

Fig. 12 Comparison of $Cost(Alg)$ with different proportion of ΔV.

图 12 增加不同比例节点的计算量比较

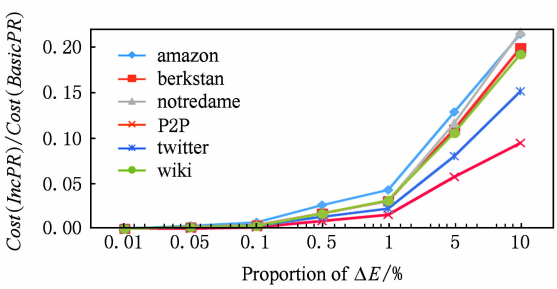

Fig. 13 Comparison of $Cost(Alg)$ with different proportion of ΔE.

图 13 增加不同比例边的计算量比较

图 14 反映了增加固定数量的边的情况下,IncPR 的计算量的变化趋势,不同数据集上的多组实验反

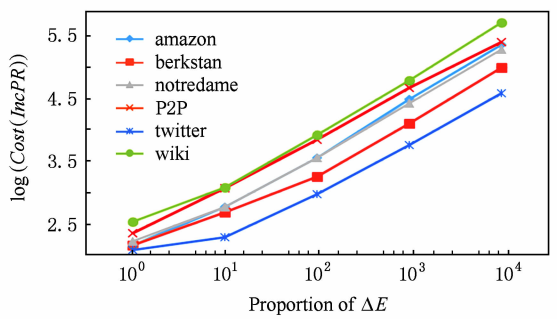

Fig. 14 Relationship of $Cost(IncPR)$ and the number of ΔE.

图 14 增加边的数量与增量算法计算量的关系

映出相似的规律,即计算量和边的变化量基本符合线性正比例关系,与理论分析得到的算法的时间复杂度大致相符.

6 总 结

旨在降低频繁变化的海量数据集上反复计算 PageRank 带来的资源消耗,本文提出了一种基于增量计算思想的并行 PageRank 算法——IncPR. IncPR 在时间复杂度上优于已有的相关算法,且不引入额外的存储开销. 理论分析及基于真实图数据的大量分布式计算实验均验证了 IncPR 的正确性和计算效率,在保证结果精度的前提下,较之于已有的增量 PageRank 算法和非增量的 PageRank 算法,IncPR 能够显著地降低计算量. 此外,IncPR 的设计思想也可以用于改进基于蒙特卡罗模拟技术的其他算法(如个性化 PageRank、单源最短路径算法等),这也将有利于降低此类应用问题的计算成本.

参 考 文 献

[1] China Computer Federation Expert Committee of Big Data. White Paper on Big Data and Industry Development of China 2013 [M/OL]. Beijing: China Computer Federation, 2013. [2016-03-20]. http://www.ccf.org.cn/sites/ccf/ccfziliao.jsp?contentId=2774793649105 (in Chinese)
(中国计算机学会大数据专家委员会. 中国大数据与产业发展白皮书 2013 [M/OL]. 北京: 中国计算机学会, 2013. [2016-03-20]. http://www.ccf.org.cn/sites/ccf/ccfziliao.jsp?contentId=2774793649105)

[2] Google Inc. Google inside search-google [EB/OL]. [2016-03-20]. https://www.google.com/intl/ta/insidesearch/howsearchworks/thestory/index.html

[3] Kunder M. The size of the world wide Web: Estimated size of Google's index [EB/OL]. [2016-03-20]. http://www.worldwidewebsize.com/

[4] World Wide Web Foundation. Internet live stats [EB/OL]. [2016-03-20]. http://www.internetlivestats.com/

[5] Page L, Brin S, Motwani R, et al. The PageRank citation ranking: Bringing order to the Web, 422 [R]. Stanford, CA: Stanford InfoLab, 1999

[6] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107-113

[7] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster computing with working sets [C] //Proc of the 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud). Berkeley, CA: USENIX Association, 2010: 10-10

[8] Malewicz G, Austern M H, Bik A J C, et al. Pregel: A system for large-scale graph processing [C] //Proc of the 2010 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2010: 135-146

[9] Gleich D, Zhukov L, Berkhin P. Fast parallel PageRank: A linear system approach, 38 [R]. Berkeley, CA: Yahoo! Research, 2004

[10] Bahmani B, Chakrabarti K, Xin D. Fast personalized pagerank on mapreduce [C] //Proc of the 2011 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2011: 973-984

[11] Sarma A D, Molla A R, Pandurangan G, et al. Fast distributed pagerank computation [J]. Theoretical Computer Science, 2015, 561 (Part B): 113-121

[12] 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, 2007, 45(2): 890-904

[13] Apache Software Foundation. Apache Hama [EB/OL]. [2016-03-20]. https://hama.apache.org/

[14] Ohsaka N, Maehara T, Kawarabayashi K. Efficient PageRank tracking in evolving networks [C] //Proc of the 21st ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2015: 875-884

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

[16] Sarma A D, Gollapudi S, Panigrahy R. Estimating PageRank on graph streams [J]. Journal of the ACM, 2011, 58(3): 1165-1182

[17] Sarlós T, Benczúr A A, Csalogány K, et al. To Randomize or not to randomize: Space optimal summaries for hyperlink analysis [C] //Proc of the 15th Int Conf on World Wide Web. New York: ACM, 2006: 297-306

[18] Bahmani B, Chowdhury A, Goel A. Fast incremental and personalized pagerank [J]. Proceedings of the VLDB Endowment, 2010, 4(3): 173-184

[19] Kamvar S D, Haveliwala T H, Manning C D, et al. Extrapolation methods for accelerating PageRank computations [C] //Proc of the 12th Int Conf on World Wide Web. New York: ACM, 2003: 261-270

[20] Kamvar S, Haveliwala T, Manning C, et al. Exploiting the block structure of the Web for computing pagerank, 579 [R]. Stanford, CA: Stanford InfoLab, 2003

[21] Langville A, Meyer C. Updating PageRank using the group inverse and stochastic complementation, CRSC-TR02-32 [R]. Raleigh: Mathematics Department, North Carolina State University, 2002

[22] Langville A N, Meyer C D. Updating pagerank with iterative aggregation [C] //Proc of the 13th Int World Wide Web Conf on Alternate Track Papers & Posters. New York: ACM, 2004: 392-393

[23] Desikan P, Pathak N, Srivastava J, et al. Incremental PageRank computation on evolving graphs [C] //Special Interest Tracks and Posters of the 14th Int Conf on World Wide Web. New York: ACM, 2005: 1094-1095

[24] Lofgren P. On the complexity of the Monte Carlo method for incremental PageRank [J]. Information Processing Letters, 2014, 114(3): 104-106

[25] Tong H, Papadimitriou S, Philip S Y, et al. Proximity tracking on time-evolving bipartite graphs [C] //Proc of the 8th SIAM Int Conf on Data Mining. Philadelphia: Society for Industrial and Applied Mathematics Publications, 2008: 704-715

[26] Mcsherry F. A uniform approach to accelerated PageRank computation [C] // Proc of the 14th Int Conf on World Wide Web. New York: ACM, 2005: 575-582

[27] Seo S, Yoon E J, Kim J, et al. Hama: An efficient matrix computation with the mapreduce framework [C] //Proc of the 2nd IEEE Int Conf on Cloud Computing Technology and Science, CloudCom. Piscataway, NJ: IEEE, 2010: 721-726

[28] Jure Leskovec. Stanford large network dataset collection [EB/OL]. [2016-03-20]. http://snap.stanford.edu/data/index.html

Jiang Shuangshuang, born in 1990. Master. Her main research interests include distributed computing, incremental computing and data mining.

Liao Qun, born in 1987. PhD candidate. His main research interests include distributed computing, task scheduling and data mining.

Yang Yulu, born in 1961. Professor and PhD supervisor in Nankai University. His main research interests include interconnection network, parallel and distributed processing.

Li Tao, born in 1977. PhD and associate professor in Nankai University. Member of the ACM and the IEEE Computer Society, Senior member of China Computer Federation. His main research interests include parallel and distributed processing, high-performance computing and network information mining.