分布式环境下基于马尔科夫链的图流三角近似计算

金宏桥,董一鸿,陈华辉,钱江波
(宁波大学信息科学与工程学院,浙江宁波 315211)

摘 要: 图三角数量的计算是计算网络聚集系数和传递性的重要步骤.大数据背景下,以采样为策略的近似计算成为图三角计算的主要方法,然而此类方法面临时空消耗和计算错误性两大难题.本文提出了一种针对图流的基于马尔科夫链的图三角近似计算算法,该算法以窗口作为图流处理单位,将马尔科夫链与采样相结合,保证降低错误率的同时实现动态适应内存空间的变化.实验显示,相较其他三角形近似计算算法,该算法在错误率上降低2~4倍,时间消耗上也有很大改进.

关键词: 图三角;图流;马尔科夫链;大数据;Spark
中图分类号: TP391 **文献标识码:** A **文章编号:** 0372-2112 (2018)09-2139-010
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.09.014

DATC-MC:Distributed Approximate Triangle Counting Based on Markov Chain in Graph Streaming

JIN Hong-qiao,DONG Yi-hong,CHEN Hua-hui,QIAN Jiang-bo
(*Faculty of Electrical Engineering and Computer Science,Ningbo University,Ningbo,Zhejiang* 315100,*China*)

Abstract: Counting triangles in a graph is an important step to calculate the clustering coefficient and the transitivity ratio of the network. Under the background of big data,the sampling method is the main method used in triangle counting. However,two problems of spatiotemporal costs and computing errors are very difficult. An approximate triangle counting algorithm based on Markov chain in the graph stream is proposed. Treating the window as processing unit of graph stream,it uses Markov chain to implement dynamic adaptation of memory and to reduce error ratio while sampling. Compared with other algorithms,the proposed algorithm can drop the error rate by 2~4 times,and the time consumption has a great improvement.

Key words: triangle;graph streaming;Markov chain;big data;spark

1 引言

近年来,大规模数据的网络分析越来越受到关注,如万维网、P2P(Peer-to-peer)网络和社交网络等,一般可以用含有特定信息的图作为它们的模型.图中的三角形是复杂网络分析的重要角色,通过三角形的分布可以区分哪些是垃圾邮件的主人;角色行为识别中,通过使用者参与的三角形的数量可以判断这个使用者的地位;生物信息学中的主题检测需要计算三元组的频率;三角形巨大的数量可以与蛋白质交互网络的拓扑结构和功能性相联系.

目前图三角形的计算主要分为准确计算(exact-counting)和近似计算(approximate-counting).然而,大图的准确计算规模大,计算的时空消耗大,外存算法的I/O消耗也大.研究人员关注如何减少时空消耗和I/O次数.近似计算比准确计算空间消耗少,具有更广泛的应用前景.近年来,图三角近似计算的准确性逐步提高,可以代替准确计算,引起研究人员的极大兴趣.大部分学者将近似计算的重点放在采样上,然而单纯的采样方法不稳定,不能确保稳定的错误率,不能使内存空间达到最优,也不能针对不同速度的流做出不同的反应,尤其在应对极高速度的流时,单纯的顺序计算不能做

收稿日期:2016-10-25;修回日期:2017-09-22;责任编辑:梅志强
基金项目:国家自然科学基金(No.61472194,No.61572266);浙江省自然科学基金(No.Y16F020003);宁波市自然科学基金资助项目(No.2017A610114)

出快速的实时响应.

针对传统近似计算算法的不足,本文提出了基于马尔科夫链的图流三角分布式近似计算(Distributed Approximate Triangle Counting based on Markov Chain in graphstreaming,DATC-MC)算法. 以窗口为单元进行三角计数,可应对各种速度的图流. 在采样过程中引入马尔科夫链的思想,允许采样概率根据上一时刻的概率变化来适应空间变化降低错误率. 本文在分布式环境下实现了 DATC-MC 算法,既保证了算法的错误率足够低,又充分的利用并行计算减少时间消耗.

本文的主要贡献如下:(1)提出了大规模图流三角计算算法 DATC-MC,采用窗口为单元进行计算,将马尔科夫链与采样结合,充分利用内存,算法错误率显著下降;(2)将顺序流的图三角计算并行化,首次提出了基于分布式平台 Spark 的图流三角计算方法,时间复杂度降低;(3)通过实验将 DATC-MC 算法和其他算法进行对比,实验结果显示了 DATC-MC 算法的有效性和可行性.

2 相关工作

最早的三角形计算算法是静态图中的点边迭代准确算法. 1997 年,Alon 等人将点迭代和矩阵相乘结合在一起得到 AYZ[1] 算法. 此后依次出现了 Node-iterator-core[2]、Forward-hashed[3]、Compact-forward[4] 等改进的准确算法. 2011 年 Chu 和 Cheng[5] 提出的基于图划分的外存图三角准确计算算法. 随着分布式框架的应用,2013 年相继出现了基于 MapReduce 框架的 GP[6]、TTP[7] 以及 CTTP[8] 算法,将三角形分类,一定程度上减少了时间消耗. 同时国内,2013 年 XIAOCHENG HU 和 YUFEI TAO[9] 设计了一种将无向图转化为有向图计算的图三角算法. 伴随着准确计算的发展,近似计算也在相应发展,2010 年,分别出现了基于度对顶点划分和随机矩阵迹的近似算法.

早期的图流算法模型是在流中近似计算图三角,2005 年 Hossein Jowhari[10] 和 2006 年 Luciana S. Buriol[11] 提出的单、多通道算法,通过设置估计器来估计图三角的数量. 2013 年 A. Pavany[12] 等人设计了一个基于邻居采样的图流算法. 从流中随机采样一条边,然后采样和该边有共同顶点的边,其中设置了数个估计器. 2015 年 LIM Y 等人提出了基于边采样的图流算法 MASCOT[13],仅对边进行采样,使用了 DOULION[14] 方法估计图三角数量,有效使用了内存. 同年 Laurent Bulteau[15] 等人设计了一个基于 2-path 的方法来估计图流三角形的个数,算法设置了多个稀疏图,通过稀疏图与原图的传递系数关系估计图三角的数量. 图流算法与静态算法相比,更适用于在线计算.

3 基本概念

定义 1 图三角[9] 给定一个图 $G=(V,E)$,它包含了一个顶点集合 V、一个边的集合 E,$|V|$ 和 $|E|$ 分别表示顶点和边的数量. 如果顶点 u、v、$w\in V$,边 $\{u,v\}$、$\{v,w\}$、$\{w,u\}\in E$,那么 3 个顶点和 3 个边组成一个三角形,称为 Δ_{uvw}.

定义 2 图三角实际数量和近似数量[14] 图三角实际数量是图 G 中三角形的真实数量,用 Δ 表示;图三角的近似数量是通过近似算法计算得到的图 G 中三角形的数量,用 T 表示.

定义 3 度和邻域[9] 顶点 v 的邻域 $\Gamma(v)$ 表示所有与顶点 v 相邻的点,满足 $\Gamma(v)=\{u\in V:\{u,v\}\in E\}$. 顶点 v 的度 $d(v)$ 是顶点 v 连接的所有边的数量或者是它的邻域,满足 $d(v)=|\Gamma(v)|$. 图 G 最大的度 $d_{\max}(G)$ 是指图中顶点最大的度,即 $d_{\max}(G)=\max\{d(v):v\in V\}$. m、n 分别表示图中边、点的个数,顶点的度和图中的边数满足 $\sum_{v\in V}d(v)=2m$.

定义 4 图流 图流指图中的边是以流的形式加入或删除,如 $\{\{u,v;+/-1\}、\{v,w;+/-1\}、\{w,u;+/-1\}\cdots\}$,其中 $+1$ 表示当前边是插入边,-1 表示当前边是删除边.

定义 5 图流三角计算 在形为 $\{\{u,v;+/-1\}、\{v,w;+/-1\}、\{w,u;+/-1\}\cdots\}$ 的图流中,在不同时刻计算已到达的边组成的图三角形的数量.

表 1 列出了本文所用的符号与对应的描述.

表 1 符号表

符号	描述
G	图
V	顶点集合
E	边集合
$d_{\max}$	最大的度
p_i	窗口采样概率
η_i	稀疏图采样概率
W_i^+	窗口中插入边集合
W_i^-	窗口中删除边集合
t_i	第 i 时刻

4 基于马尔科夫链的图流三角计算

4.1 马尔科夫链

马尔可夫链(Markov Chain)[16],描述了一种状态序列 $X_1,X_2,X_3\cdots\cdots$,其每个状态值取决于前面有限个状态. 马尔可夫链是具有马尔可夫性质的随机变量的数列. 这些变量的范围,即它们所有可能取值的集合,被称为“状态空间”,而 X_n 的值则是在时间 n 的状态. 如果 X_{n+1} 对于过去状态的条件概率分布仅是 X_n 的函数,则

$$P(X_{n+1}=x|X_1=x_1,X_2=x_2,\cdots,X_n=x_n)=P(X_{n+1}=x|X_n=x_n) \quad (1)$$

4.2 DATC-MC 算法的理论依据

引理 1[14]　设图 G 经过概率 p 均匀采样后得 G',G' 中三角形的数量为 Δ',那么 $\mathrm{E}(\Delta'/p^3)=\Delta$.

定理 1　设对原图 G 以概率 p_1 抽样得图 G_1,对图 G_1 以概率 p_2 抽样得图 G_2,……对图 G_{m-1} 以概率 p_m 抽样得图 G_m,图 G_m 中的三角形个数为 Δ_m,那么

$$\mathrm{E}\left[\Delta_m\Big/\Big(\prod_{i=1}^{m}p_i\Big)^3\right]=\Delta$$

若对原图 G 以概率 p'抽样得 G',其中 $p'=\prod_{i=1}^{m}p_i$,那么

$$E[\Delta_m/(p')^3]=\Delta$$

证明　对于真实三角形数量为 Δ 的图 G,如果第一次以概率 p_1 采样后得到图 G_1,那么图 G_1 对图 G 的三角形数量估计值 T_1 的期望为 $\mathrm{E}[T_1]=\Delta$,假设图 G_1 中的三角形数量为 Δ_1,根据引理 1,$\mathrm{E}[\Delta_1/p_1^3]=\Delta$. 第二次以概率 p_2 采样图 G_1 后得到图 G_2,那么图 G_2 对图 G_1 的三角形数量估计值 T_2 的期望为 $\mathrm{E}[T_2]=\Delta_1$,假设图 G_2 中的三角形数量为 Δ_2,那么 $\mathrm{E}[\Delta_2/p_2^3]=\Delta_1$. 同理,第 m 次以概率 p_m 采样图 G_{m-1} 后得到图 G_m,那么图 G_m 对图 G_{m-1} 的三角形数量估计值 T_m 的期望为 $\mathrm{E}[T_m]=\Delta_{m-1}$,假设图 G_m 中的三角形数量为 Δ_m,那么

$$\mathrm{E}[\Delta_m/p_m^3]=\Delta_{m-1}$$

由于

$$\begin{aligned}\Delta&=\mathrm{E}[\Delta_1/p_1^3]=\mathrm{E}[\mathrm{E}[\Delta_2/p_2^3]/p_1^3]\\&=\mathrm{E}[\mathrm{E}[\Delta_2]/(p_1p_2)^3]\\&=\mathrm{E}[\Delta_2/(p_1p_2)^3]=\cdots\\&=\mathrm{E}\left[\mathrm{E}[\Delta_m]\Big/\Big(\prod_{i=1}^{m}p_i\Big)^3\right]\\&=E\left[\Delta_m\Big/\Big(\prod_{i=1}^{m}p_i\Big)^3\right]\end{aligned}$$

同时,如果以概率 p'(其中 $p=\prod_{i=1}^{m}p_i$)采样后得到图 G',图 G'对图 G 的三角形数量的估计值 T'的期望为 $\mathrm{E}[T']=\Delta$,假设图 G'中的三角形数量为 Δ',$\mathrm{E}[\Delta'/p'^3]=\Delta$. 综上,图 G 依次经过概率 p_1、p_2、…、p_m 采样得到的图 G_m 中三角形的数量除以 $\prod_{i=1}^{m}p_i$ 的三次方后的值估算图 G 中三角形的数量等同于用图 G 经过概率 p'(其中 $p=\prod_{i=1}^{m}p_i$)采样得到的图 G'中三角形的数量除以 p'的三次方后的值估算图 G 中三角形的数量.

假设在原图 G 中,给每一个三角形附加一个变量 δ_i,$i=1\cdots\Delta$(Δ 为三角形总数),如果在 G_m 中,第 i 个三角形是存在的,那么 δ_i 赋值 1,不存在则赋值 0. 由于两个三角形可能共有一条边,所以三角形的附加变量不是独立的. 所以方差为:

$$\begin{aligned}&\mathrm{Var}(T)\\&=\mathrm{Var}\left(\frac{1}{\Big(\prod_{i=1}^{m}p_i\Big)^3}\sum_{i=1}^{\Delta}\delta_i\right)\\&=\frac{1}{\Big(\prod_{i=1}^{m}p_i\Big)^6}\sum_{i=1}^{\Delta}\sum_{j=1}^{\Delta}\mathrm{Cov}(\delta_i,\delta_j)\\&=\frac{1}{\Big(\prod_{i=1}^{m}p_i\Big)^6}\Big(\Delta\Big(\Big(\prod_{i=1}^{m}p_i\Big)^3-\Big(\prod_{i=1}^{m}p_i\Big)^6\Big)\\&\quad+2k\Big(\Big(\prod_{i=1}^{m}p_i\Big)^5-\Big(\prod_{i=1}^{m}p_i\Big)^6\Big)\Big)\end{aligned}$$

令 $p=\prod_{i=1}^{m}p_i$,$\mathrm{Var}(T)=\frac{1}{p'^6}(\Delta(p'^3-p'^6)+2k(p'^5-p'^6))$. 其中 k 表示有多少对三角形共边. 由 $\mathrm{Var}(T)$ 可知,p'的值越大,方差越小. 根据切比雪夫不等式:

$$P(|X-\mathrm{E}(X)|\geqslant\varepsilon)\leqslant\frac{D(X)}{\varepsilon^2}$$

可知:$P(|T-\Delta|\geqslant\varepsilon\Delta)\leqslant\frac{(p^3-p^6)}{p^6\varepsilon^2\Delta}+2k\frac{(p^5-p^6)}{p^6\varepsilon^2\Delta^2}$[16].

定理 2　任意时刻,窗口中删除边当且仅当以概率 $p=1$ 采样更新稀疏图,才能得到当前图流均匀采样的稀疏图.

证明　**充分性**　反证法,假设在时刻 t_i,图流到达的真实图为 G_i,G_i 中的边为(1,2)、(1,3)、(2,3)、(2,4),稀疏图的阈值为 3,其中保留的边为(1,2)、(1,3)、(2,3). 在时刻 t_{i+1},到达的边只有删除边,且为(2,3),如果删除边以概率 $p<1$ 采样,那么有可能删除边(2,3)被舍去不用于更新稀疏图,那么稀疏图仍然是(1,2)、(1,3)、(2,3). 但此时真实图 G_{i+1}为(1,2)、(1,3)、(2,4),稀疏图不是当前图流均匀采样后的图. 由此可知,窗口中的删除边需以概率 $p=1$ 采样下来用于更新稀疏图,使稀疏图成为当前图流均匀采样后的图.

必要性　假设在时刻 t_i,图流到达的真实图为 G_i,G_i 中的边为(1,2)、(1,3)、(2,3)、(2,4),稀疏图的阈值为 3,其中保留的边为(1,2)、(1,3)、(2,3). 在时刻 t_{i+1},到达的边为删除边 (2,3),只有保证稀疏图中的边为(1,2)、(1,3)时,才能使稀疏图成为当前图流均匀采样后的图. 若需保证稀疏图的边为(1,2)、(1,3),那么删除边一定是以概率 $p=1$ 采样下来用于更新稀疏图.

5 DATC-MC 算法描述

串行计算图三角开销很大,恰当的利用分布式平台能提高效率. 稀疏图和窗口均保存在 RDD 中,将窗口中的边涉及到的三角形计算分散到各个节点. 并行化减少每条边的平均计算时间,提高效率. 为达到并行计算,本文设计了分布式算法,并分析了时间复杂度.

5.1 窗口、稀疏图存储格式

窗口和图的存储格式如图1所示,窗口中的边是图流的输入形式,窗口中的边数根据时间来切割. 窗口是多个格式为$(u,v;+/-1)$的边组成的列表,其中u、v是边的两个顶点,$+/-1$表示此边是插入边或删除边. 稀疏图是由数对(key,value)组成,其中key是顶点,value中保存的是顶点的邻点的列表. 邻点也用(key,value)表示,其中key表示邻点,value值表示顶点与邻点之间的边的当前状态,value值为+1表示该边在当前时刻之前被插入;值为+2表示该边当前时刻被插入;值为0表示该边当前时刻被删除.

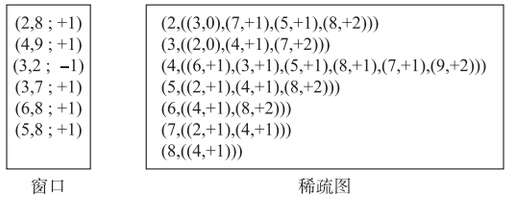


图1 窗口(左)和稀疏图(右)的存储格式

5.2 处理插入边和删除边

窗口中的边有插入边和删除边,它们是无序的. 在采样之前,窗口进行一次预处理,将重复插入删除的边抵消,并更新稀疏图. 具体步骤如下:

步骤1 将窗口中的key-value对分配到各个节点并行的进行map操作,使每一个$(u,v;+/-1)$中$u<v$,输出以(u,v)为key,$(+/-1)$为value.

步骤2 进行reduceByKey操作,有同样key的value相加,结果为0表示此边插入删除的次数相同,可以抵消;其他结果只有+1和−1,表示最终此边是插入或删除,将value结果为0的过滤掉. 接着将其中所有标为+1的边经过当前的概率采样(5.4节将说明采样过程).

步骤3 最后将每一个$(u,v;+/-1)$进行flatMap操作,输出$(u,(v,+/-1)),(v,(u,+/-1))$,再与稀疏图进行union操作,接着对稀疏图进行reduceByKey操作,将与稀疏图进行union操作的窗口中的边加入稀疏图中的每一个顶点的邻点集中. 如图2,稀疏图中,新加入的边没有对应的顶点时,会增加相应的顶点及其邻点,如(1,(2,+2)),有对应的顶点时,在原来的邻点集中添加新的邻点,如(2,((3,+2),(7,+1),(1,+2)))中的(3,+2)和(1,+2);减去的边,如果有对应的顶点和邻点,就与存在的值相加,得0,如(3,((2,+2),(4,+1),(7,0)))中的(7,0),没有对应的顶点或邻点时,就添加对应的顶点和邻点,也标为0.

5.3 采样后图三角的计算

稀疏图中图三角数量需要准确计算,算法从分配唯一标志、去除重复计算两方面达到对当前稀疏图中三角形数量的准确计算.

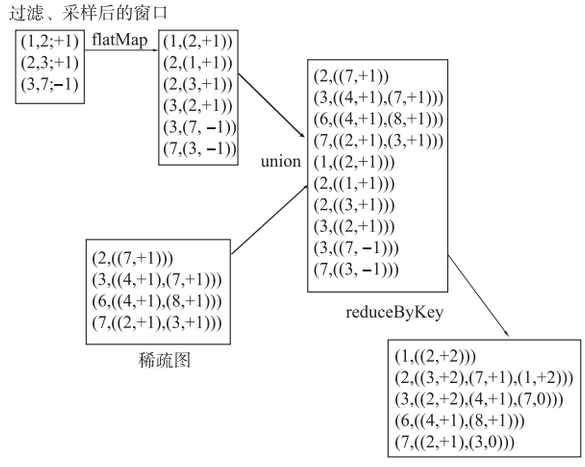


图2 窗口操作

(1)分配唯一标志

对于窗口中每一条边,首先并行地将每一个$(u,v;+/-1)$执行flatMap操作,给其中的u,v分配标志flag,输出$(u,\text{flag}+\text{""}+\text{"}+/-1\text{"})$和$(v,\text{flag}+\text{""}+\text{"}+/-1\text{"})$. 将输出结果广播到各个节点,然后与稀疏图进行join操作,相同的key合并为$(u,(\text{flag}+\text{""}+\text{"}+/-1\text{"},((N_1,+1),(N_2,+1),\cdots,(N_d+1))))$. 然后进行map操作,将flag值调到key的位置,再进行reduceByKey操作,将相同的flag的value放在一起,即将窗口中的同一条边的两个顶点的邻点放在一起去求交集. 由于多个顶点求交集是分布在各个计算节点上,减少了平均每条流入边求邻点交集的计算时间. 具体过程见图3.

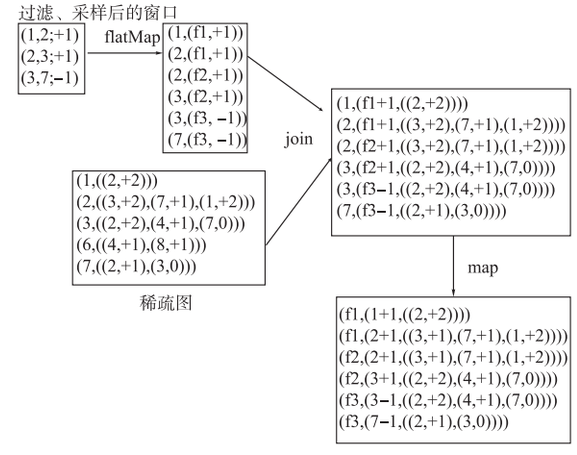


图3 flag设置

(2)去除重复计算

当前时刻稀疏图里对三角形的计算是准确计算,当前时刻的三角形数量是由上一时刻三角形的数量加上或减去当前流入的窗口造成三角形数量增加或减少的数量. 即:

$$C_i=\begin{cases}C_i^{\mathrm{W}}, & i=1\\ C_{i-1}+C_i^{\mathrm{W}}, & i\neq 1\end{cases}\tag{2}$$

其中,C_i^{W} 表示当前时刻 t_i,窗口的流入造成三角形数量的增量或减量,C_i 表示当前时刻稀疏图中的三角形数量.

上述求出具有相同 flag 的邻点集交集数量并不能简单作为当前增加或减少的三角形数量. 如图 4 所示,在同时增加或减少几条边时,会形成如下 9 种情况的三角形,这 9 种三角形由计算过程中每个边和邻点后的“0”、“+2”、“+1”区分. 其中,图 4(*b*)和(*e*)的三角形被多增加或减少了 1 次;图 4(*c*)和(*f*)的三角形被多增加或减少了 2 次;图 4(*g*)和(*i*)的三角形被多减少或增加了 1 次. 需要将重复增加(减少)的三角形数量减去(加上).

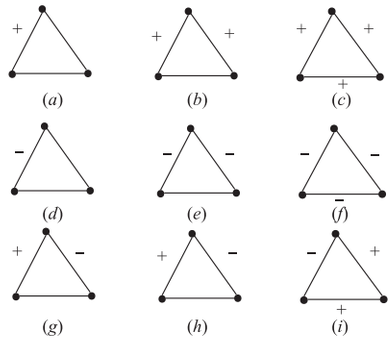


图4　9种三角形

5.4　基于马尔科夫链的采样过程

(1)窗口计算单元

现有的图流算法中对于图流中的每一条边的到来进行一次三角形计算,这种依次计算的方法对于高速图流并不适用,尤其对于单条边计算时间大于两条边到达间隔时间的情况. 如在边 e_1 在 t_i 时刻到达,边 e_2 在 t_{i+1}时刻到达. 边 e_n 在 t_{i+n-1}时刻到达,边 e_1 计算三角形的时间为 t'时,$t'>t_{i+n-1}-t_i$. 如图 5 所示的窗口模型,将短时间内到达的数条边放入一个窗口中,将整个窗口中的边用来进行一次三角形计算,整个窗口中的边称为计算单元,即在时刻 t_i,窗口 W_i 中含有多条边,时刻是人为规定的,不同时刻间隔一定的小时间段,不同时刻的边数量是随机的. 同一窗口里的边有插入边和删除边,窗口 W_i 中插入边记做 W_i^+,删除边记做 W_i^-.

(2)稀疏图阈值

为保证运算速度,在内存中保持一个当前最大的稀疏图 S_i(大小为$|S_i|$)保存原图的主要信息. 对原图以均匀采样的方式作为稀疏图存入内存. 稀疏图越大,越接近原图,计算三角形越有效. 设置稀疏图阈值$|M_i|$监测内存的变化,当可使用的内存变大时,稀疏图阈值随之增大,反之随之减小. 稀疏图的大小尽可能接近稀疏图阈值,但不超过稀疏图阈值,即$|S_i|\leqslant|M_i|$.

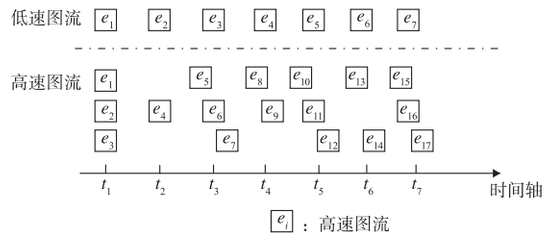


图5　低速图流(上)与高速图流(下)

(3)采样过程

采样概率 p 设置太大内存不够,设置太小方差增大. 为了充分利用内存,降低估算三角形数量的错误率,采用动态采样概率的方法. 在图流动态流入的过程中,设置一个可以随内存大小变化且尽可能大的概率去采样,使计算三角形的错误率尽可能低.

根据定理 1,对已流入的边多次采样,对新来的边一次采样,分别使用不同的概率,利用概率值之间的关系达到已流入的和新流入的边同等程度的采样,最终达到对全图流的均匀采样. 为了充分利用内存空间,提出了基于马尔科夫链的采样概率,不同时刻窗口中的边的采样概率不同,当前时刻的采样概率只与上一时刻采样的概率有关而与上一时刻之前的时刻采样的概率无关. 具体步骤如下:

步骤 1　在第一个时刻窗口到来之前,内存中没有图,稀疏图 S_1 的大小为 0. 在第一个时刻窗口到来时(通常情况下,窗口的大小远比稀疏图小),窗口中的边直接(即插入边以概率 $p_1=1$,删除边 $p_{\mathrm{delete}}=1$,根据定理 2)加入稀疏图后未达到稀疏图的阈值$|M_1|$,即$|S_1-W_1^-+W_1^+|<|M_1|$. 窗口中的边流入稀疏图后,稀疏图 S_1',以概率 $\eta_1=1$ 采样下来,即全部留下,成为 S_2.

步骤 2　一段时间后,在时刻 t_i,窗口中的边 W_i 直接(即插入边以概率 $p_1=1$,删除边 $p_{\mathrm{delete}}=1$)加入稀疏图 S_i,稀疏图的大小$|S'_i|=|S_i-W_i^-+W_i^+|$超过阈值$|M_i|$,即$|S'_i|>|M_i|$. 稀疏图以 $\eta_i=|M_i|/|S'_i|$的概率采样出 S'_i 的一部分边保留下来,其余舍去,使稀疏图的大小$|S'_i|$趋于$|M_i|$值,成为 S_{i+1}.

步骤 3　在下一时刻 t_{i+1},窗口 W_{i+1}中的边 W_{i+1}^+到来时需经过一次概率为 $p_{i+1}=p_i\cdot\eta_i$ 的采样之后,再加入稀疏图,其中删除边 W_{i+1}^-直接加入(删除相应已有的边). 对于加入过后的稀疏图 S'_{i+1},如果$|S'_{i+1}|>|M_{i+1}|$,稀疏图以 $\eta_{i+1}=|M_{i+1}|/|S'_{i+1}|$的概率采样出$S'_{i+1}$的一部分边保留下来,使稀疏图的大小$|S'_{i+1}|$趋于$|M_{i+1}|$值,成为 S_{i+2};如果$|S'_{i+1}|<|M_{i+1}|$,稀疏图以 $\eta_{i+1}=1$ 的概率,即将稀疏图全部保留下来,成为 S_{i+2}.

上述过程中概率 p_i 仅仅与前一时刻的概率 p_{i-1}有

关,与前一时刻之前的 p_{i-2} 至 p_1 无关,为基于马尔科夫链的采样概率. 在时刻 t_i,窗口的采样概率 p_i 满足如下公式,η_i 为稀疏图的采样概率:

$$p_i=\begin{cases}1, & i=1\\ p_{i-1}\cdot\eta_{i-1}, & i\neq 1\end{cases} \tag{3}$$

$$\eta_i=\begin{cases}1, & |S_{i-1}|-|W_{i-1}^-|+|W_{i-1}^+|\cdot p_{i-1}<=|M_{i-1}|\\ \dfrac{|M_{i-1}|}{|S_{i-1}|-|W_{i-1}^-|+|W_{i-1}^+|\cdot p_{i-1}}, & \\ & |S_{i-1}|-|W_{i-1}^-|+|W_{i-1}^+|\cdot p_{i-1}>|M_{i-1}|\end{cases} \tag{4}$$

在图流流入过程中,任意时刻的真实三角形数量由当前时刻稀疏图中三角形准确数量和当前概率值共同估计.

算法 1 DATC-MC

```
Input: windows W_i (a list of edges) of Graph stream and available size M_i of
       memory at time t_i.
Output: the estimated number T_i of triangle of arrived graph stream at
        time t_i.
1. while(null! = Combine(W_i)){
2.     if(1 = = i){p_i = 1}//插入边采样概率初始值
3.     if(1! = i){p_i = p_{i-1} * η_{i-1}}//插入边采样概率和上一时刻的
       关系
4.     S'_i = S_i. Add(W_i^+. Sampleby(p_i)). Delete(W_i^-)//增加经过采
       样的插入边,删除全部删除边
5.     if(S'_i. Size() > M_i. Size()){η_i = M_i. Size()/S'_i. Size()}//如
       果中间状态的稀疏图大小超过内存阈值求超出的程度
6.     if(S'_i. Size() < = M_i. Size()){η_i = 1}
7.     S_{i+1} = S'_i. Sampleby(η_i)//中间状态的稀疏图采样后得到最终
       的稀疏图
8.     T_i = S_{i+1}. Counttriangle()/(p_i * η_i)^3//求得稀疏图中三角形的
       准确数量并估计图流中的图三角数量
9.     Output ← T_i
10.    }
11.}
12. Combine(w){//去掉同一窗口中重复插入删除的边
13.    w^+, w^- ← w
14.    for(e ← w^+){
15.        if(w^-. Contain(e)){
16.        w^+. Delete(e)
17.        w^-. Delete(e)
18.      }
19.    }
20.}
21. Add(w){ //增加插入边
22.    for(e ← w){add e}
23.}
24. Delete(w){ //去除删除边
25.    for(e ← w){
26.        if(e exit){delete e}
27.        if(e not exit){}
28.    }
29.}
```

5.5 时间复杂度分析

设 n 为稀疏图中顶点数量,$d_{\max}$ 为顶点中最大的度,任一条边寻找对应顶点的时间复杂度为 $O(2n)$,求出顶点的交集的时间复杂度上界为 $O(d_{\max}^2)$,所以对于 w 条边,计算三角形的复杂度上界为 $O(w(2n+d_{\max}^2))$. 并行算法中,设并行度为 pa,寻找对应顶点和求出顶点交集的时间复杂度均降为原来的 1/pa. 为使并行有效,增加了两次对窗口、一次对窗口与稀疏图的交集和一次对稀疏图的遍历操作,时间复杂度上界为 $O((4w+nd_{\max})/\text{pa})$,所以并行算法的时间复杂度上界为 $O((w(2n+d_{\max}^2+4)+nd_{\max})/\text{pa})$. 只要并行度 pa 大于 1,并行算法的时间复杂度就低于普通流式近似算法.

6 实验

6.1 数据集和评价指标

本文实验采用集群环境为 Spark1.2.0,运行模式为 Spark On Yarn,12 台内存为 8G 的机器. 并行度在实验中具体说明. 算法 DATC-MC 使用 Scala2.10.4 开发. 实验使用了 4 个真实数据集,如表 2 所示.

表 2 实验数据集描述

Graph	Vertex Number	Edge Number	Triangle Number
DBLP	317080	1049866	2224385
YOUTUBE	1134890	2987624	3056386
ORKUT	3072441	117185083	627584181
LIVEJOURNAL	3997962	34681189	177820130

实验采用平均错误率和平均计算时间去衡量各算法的效用. 平均错误率 R_{ME} 计算公式为:

$$R_{\text{ME}}=\frac{1}{t}\sum_{i=1}^{t}\frac{|\Delta_i-T_i|}{\Delta_i} \tag{5}$$

其中,t 表示最大的时刻,Δ_i 表示第 i 时刻的准确三角形数量,T_i 表示通过算法得出的三角形近似数量. 该评价函数可有效的评价算法的性能. 平均计算时间 T_{M} 式为:

$$T_{\text{M}}=\frac{T_{\text{P}}-\sum_{i}^{t}T_{\text{R}i}}{\sum_{i}^{t}|W_i|} \tag{6}$$

其中,T_{P} 表示对整体图流的处理时间,$T_{\text{R}i}$ 表示读窗口 i 中的边所花的时间,$|W_i|$ 表示窗口 i 中所有边的数量. 平均计算时间描述的是平均每条边的计算时间,用于评估算法的时间效率.

实验将算法 DATC-MC 与 MASCOT、NAIVE 算法进行对比,来检测 DATC-MC 的性能. NAIVE 算法是基于 DOULIN 的朴素算法,和 MASCOT 算法相似,原本只支持插入流,本文先将三种算法只在插入流情况下进行比较. 然后将 NAIVE 和 MASCOT 算法稍作改变为支持

插入和删除的流,三者再一起比较. 对比实验采用 MASCOT 和 NAIVE 的 spark 版本.

6.2　仅有插入流的对比实验

本节实验只针对全部是插入边的图流,从图流的速度、稀疏图的阈值对算法的平均错误率进行对比. 图流的速度分别是一个时间单位内 20000、10000、5000 条边(记为 w = 20000/10000/5000). 稀疏图的使用空间阈值分别为整个图大小的 p 倍,p = 0.1/0.2/0.3/0.4/0.5/0.6/0.7/0.8.

图 6 ~ 9 是并行度为 10 时四个数据集上不同速度图流下的平均错误率(E_M). 由于数据集不同,各算法在不同数据集上的平均错误率也不尽相同. DATC-MC 算法在平均错误率上明显比 MASCOT 和 NAIVE 算法低. 在三种不同速度的图流下,随着 p 值的降低,三种算法的错误率差距变大,NAIVE 的平均错误率上升速度最快,DATC-MC 最慢. 尤其 p 值小于 0.2 时,三种算法的错误率明显上升. 可见当稀疏图阈值低于一定数值时,三种算法都无法保证较高的准确性. p 值为 0.1 时,三种算法平均错误率最高,本文的 DATC-MC 算法的平均错误率最小. p 值为 0.8 时,三种算法的错误率都接近于 0,均有很好的表现,其中 DATC-MC 算法仍然比其他两种算法的平均错误率低. 总体而言,DATC-MC 算法相比其他两个算法在平均错误率上降低范围在 2 ~ 4 倍. 在三种不同的图流速度下,且对于某一数据集来说,不同速度图流下平均错误率变化曲线形状较相似,表明相同的稀疏图阈值下的平均错误率相差很小,几乎不计,可见图流的速度对算法的准确性影响较小.

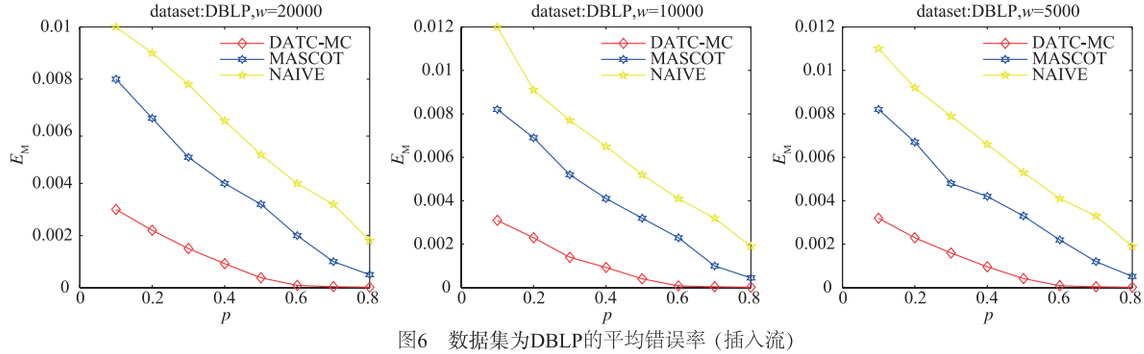


图6　数据集为DBLP的平均错误率(插入流)

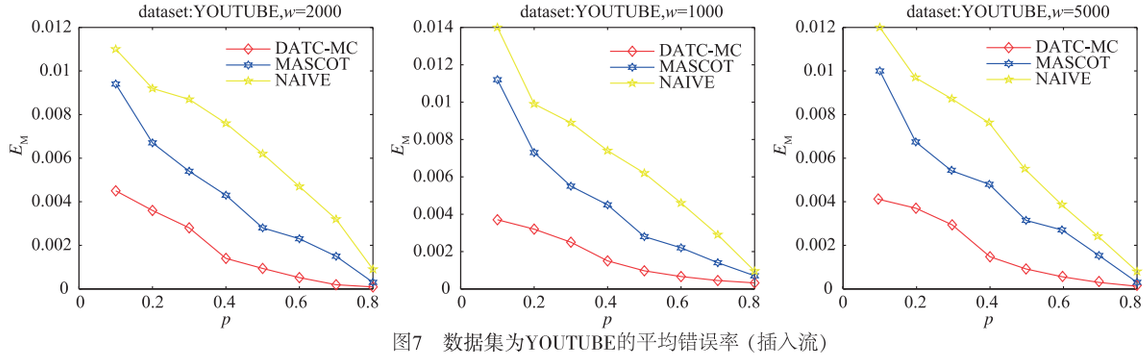


图7　数据集为YOUTUBE的平均错误率(插入流)

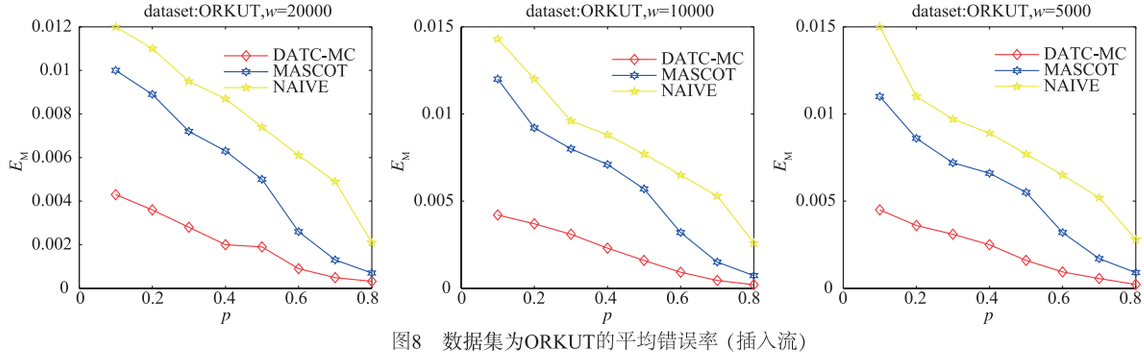


图8　数据集为ORKUT的平均错误率(插入流)

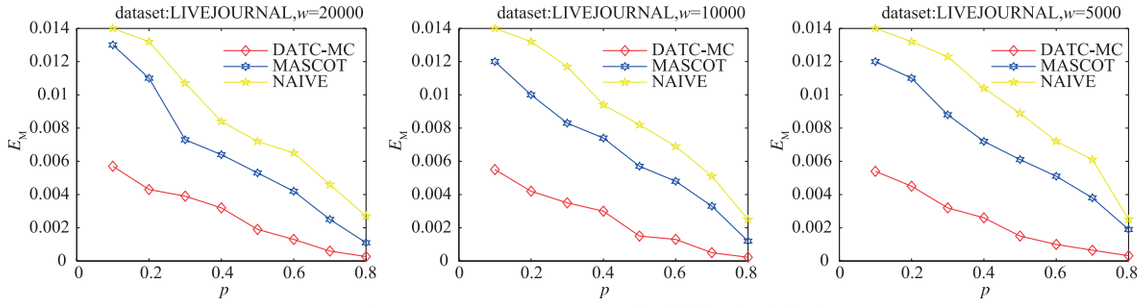


图9 数据集为LIVEJOURNAL的平均错误率（插入流）

图 10 比较了三个算法的平均计算时间. 并行度设为 10，$p=0.1$，w 分别为 20000、10000、5000 时，三个算法中，DATC-MC 算法的平均计算时间均比其他两个算法少，说明算法 DATC-MC 将过去的串行图三角计算并行化后有很高的时间效益. MASCOT 与 NAIVE 算法均为串行图流三角算法，所以两者消耗的时间更为接近. 其中稠密度较高的 ORKUT 和 LIVEJOURNAL 数据集的平均计算时间明显高于稠密度较低的 DBLP 和 YOUTUBE 数据集，表明平均计算时间与图的构成有很大的关系.

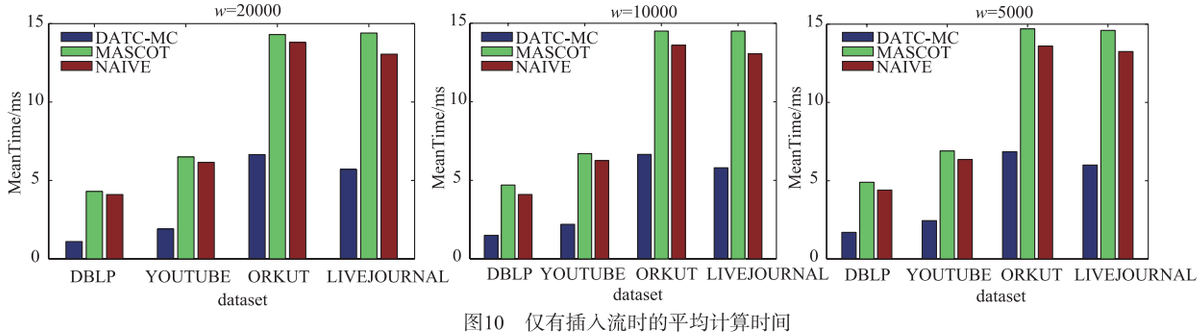


图10 仅有插入流时的平均计算时间

6.3 混合流的对比实验

本节实验采用既有插入流，又有删除流的混合流. 从图流的速度、稀疏图的阈值两个方面对算法进行对比. 图流的速度分别是一个时间单位内 20000、10000、5000 条边（记为 $w=20000/10000/5000$），稀疏图的使用空间阈值分别为整个图大小的 p 倍 $p=0.1/0.2/0.3/0.4/0.5/0.6/0.7/0.8$.

图 11 ~ 14 分别是并行度为 10 时，三种算法在四个数据集不同速度图流下的错误率. 由图可知，在混合流的情况下三种算法的表现和在仅有插入流的情况下并不相似，平均错误率均稍许增加，但三种算法的相对关系几乎不变，这表明图流的构成对算法的性能影响不大. p 值为 0.2，仍然是个转折点，DATC-MC 算法相比其他两个算法在错误率上降低范围也在 2 ~ 4 倍.

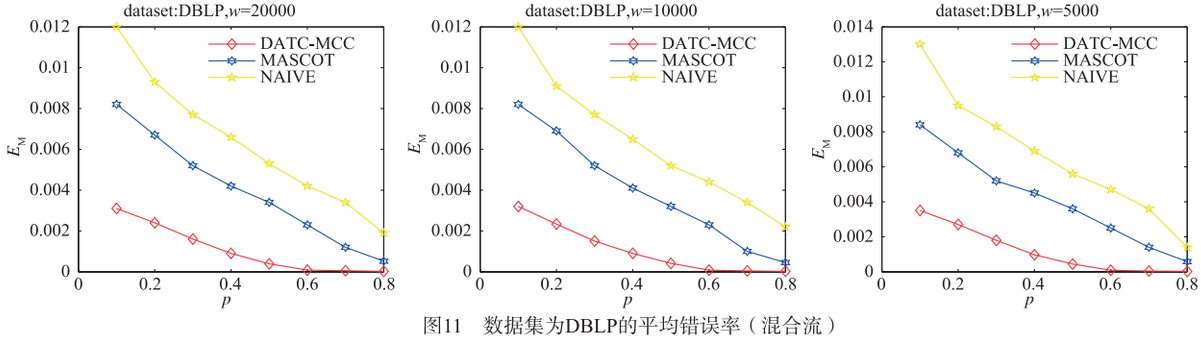


图11 数据集为DBLP的平均错误率（混合流）

6.4 DATC-MC 性能分析

通过调整并行度的大小来观察 DATC-MC 算法的运行时间. 如图 15（a），TIME RATE 是 $w=10000$ 时数据集 ORKUT 在 DATC-MC 算法上不同并行度的运行时间的比率，并行度为 1 作为基准. 为了测试算法本身的运行时间，排除图流速度的干扰，这里只包括计算三角形的运行时间，不包括图流流入的时间. 图 15（a）显示算法的运行时间随着并行度的增加而减少，且与理论

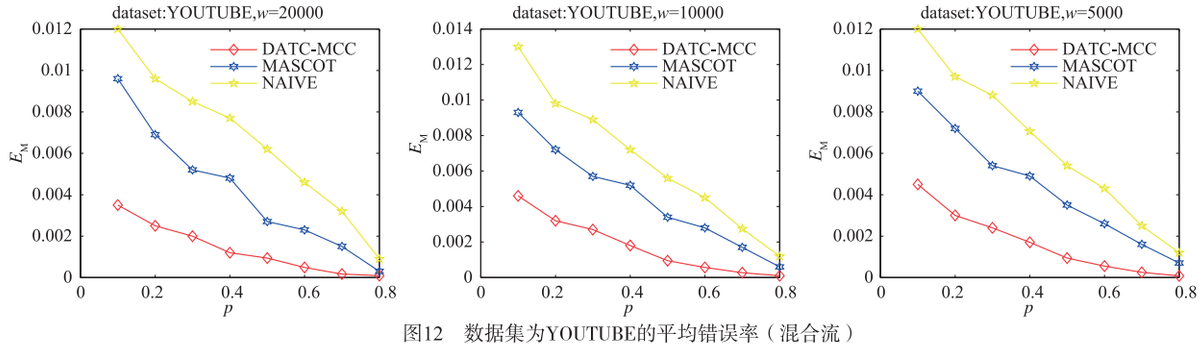


图12　数据集为YOUTUBE的平均错误率（混合流）

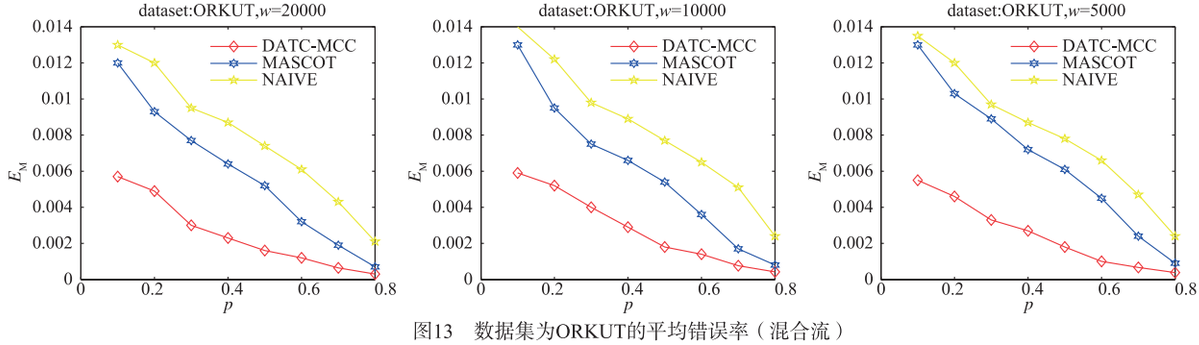


图13　数据集为ORKUT的平均错误率（混合流）

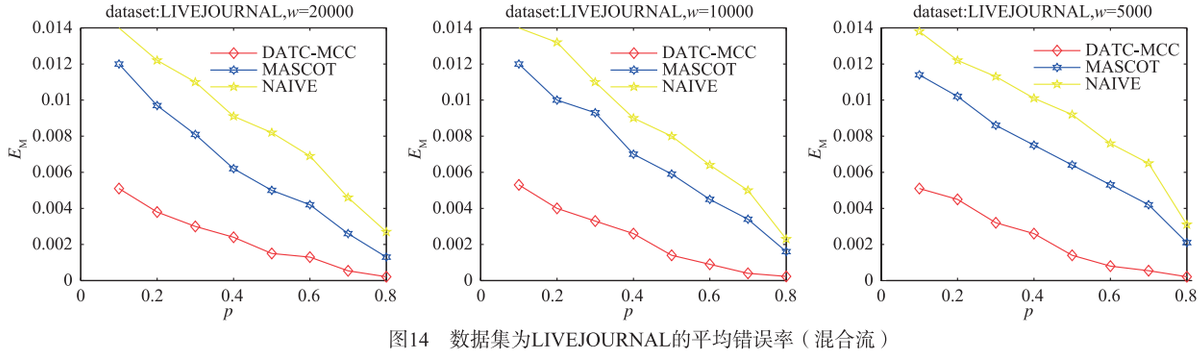
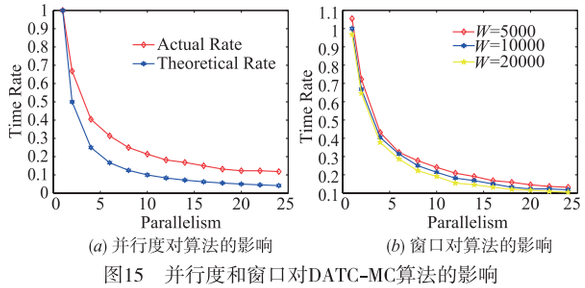


图14　数据集为LIVEJOURNAL的平均错误率（混合流）



(*a*) 并行度对算法的影响　(*b*) 窗口对算法的影响

图15　并行度和窗口对DATC-MC算法的影响

值接近. 与理论值的差距是由于各节点的通信需要消耗一定的时间. 针对不同速度的图流,即 w = 20000/10000/5000,做了算法运行时间的对比. 图 15(*b*)以使用窗口 w = 10000 为基准进行比较,速度大的图流算法的运行时间小于速度小的图流运行时间,表示 DATC-MC 算法在一定范围内更适合于高速图流.

7　结束语

近年来出现的分布式系统为图三角算法提供了新的平台. 基于分布式的三角形计算算法的可行性越来越大,逐渐成为三角形计算的研究重点. 本文提出了分布式的大规模图流三角计算算法 DATC-MC,采用窗口为单元进行计算,将马尔科夫链与采样结合,充分利用内存,算法错误率显著下降. 同时实现了对顺序流的图三角计算并行化,时间复杂度随之降低. 随着图三角近似计算在社会交互、金融交易、生物信息等领域的应用越来越广泛,降低图三角近似计算的错误率,减少计算时间依然是进一步的研究内容.

参考文献

[1] Alon N, Yuster R, Zwick U. Finding and counting given length cycles[J]. Algorithmica, 1997, 17(3): 209 – 223.

[2] Schank T, Wagner D. Finding, counting and listing all triangles in large graphs, an experimental study[J]. Lecture Notes in Computer Science, 2005, 3503: 606 – 609.
[3] Chiba N, Nishizeki T. Arboricity and subgraph listing algorithms[J]. Siam Journal on Computing, 1985, 14(1): 210 – 223.
[4] Latapy M. Main-memory triangle computations for very large (sparse (power-law)) graphs[J]. Theoretical Computer Science, 2008, 407(1 – 3): 458 – 473.
[5] Chu S, Cheng J. Triangle listing in massive networks and its applications[A]. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining[C]. San Diego, California, USA: ACM, 2011. 672 – 680.
[6] Suri S, Vassilvitskii S. Counting triangles and the curse of the last reducer[A]. International Conference on World Wide Web[C]. Hyderabad, India: ACM, 2011. 607 – 614.
[7] Park H M, Chung C W. An efficient MapReduce algorithm for counting triangles in a very large graph[A]. ACM Conference of Information and Knowledge Management[C]. San Francisco, CA, USA: ACM, 2013. 539 – 548.
[8] Park H M, Silvestri F, Kang U, et al. MapReduce triangle enumeration with guarantees[A]. ACM International Conference on Conference on Information and Knowledge Management[C]. Shanghai, China: ACM, 2014. 1739 – 1748.
[9] Hu X, Tao Y, Chung C W. I/O-efficient algorithms on triangle listing and counting[J]. Acm Transactions on Database Systems, 2014, 39(4): 1 – 30.
[10] Jowhari H, Ghodsi M. New streaming algorithms for counting triangles in graphs[J]. Lecture Notes in Computer Science, 2005, 3595: 710 – 716.
[11] Buriol L S, Frahling G, Leonardi S, et al. Counting triangles in data streams[A]. PODS′06 Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database[C]. Chicago, Illinois, USA: ACM, 2006. 253 – 262.
[12] Pavan A, Tangwongsan K, Tirthapura S, et al. Counting and sampling triangles from a graph stream[J]. Proceedings of the Vldb Endowment, 2013, 6(14): 1870 – 1881.
[13] Lim Y, Kang U. MASCOT: memory-efficient and accurate sampling for counting local triangles in graph streams[A]. 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD[C]. Sydney, NSW, Australia: ACM Press, 2015. 685 – 694.
[14] Tsourakakis C E, Kang U, Miller G L, et al. DOULION: counting triangles in massive graphs with a coin[A]. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining[C]. Paris, France: DBLP, 2009. 837 – 846.
[15] Bulteau L, Froese V, Kutzkov K, et al. Triangle counting in dynamic graph streams[J]. Algorithmica, 2016, 8503(1): 1 – 20.
[16] 姚苏,关建峰,潘华,张宏科. 基于 APT 潜伏攻击的网络可生存性模型与分析[J]. 电子学报, 2016, 44(10): 2415 – 2422.
YAO Su, GUAN Jian-feng, Pan Hua, Zhang Hong-ke. Modeling and Analysis for Network Survivability of APT Latent Attack[J]. Acta Electronica Sinica, 2016, 44(10): 2415 – 2422. (in Chinese)

作者简介

金宏桥 女,1993 年 10 月出生,安徽六安人. 现为宁波大学信息科学与工程学院硕士研究生. 主要研究方向为大数据、数据挖掘.
E-mail: nbjhqiao@163.com

董一鸿(通信作者) 男,博士,1969 年出生于浙江宁波. 宁波大学教授,主要研究方向为大数据处理、数据挖掘和人工智能等.
E-mail: dongyihong@nbu.edu.cn

陈华辉 男,博士,1964 年出身于浙江鄞州. 宁波大学教授,主要研究方向为数据库技术、流数据处理等.
E-mail: chenhuahui@nbu.edu.cn

钱江波 男,博士,1974 年出生于浙江宁波. 宁波大学教授,主要研究方向为数据库技术、流数据处理、多维数据索引技术等.
E-mail: qianjiangbo@nbu.edu.cn