计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202440120
61(8): 1882−1903, 2024

针对大规模动态图流三角形计数的边哈希分布式抽样算法

何玉林[1,2] 吴 波[2] 吴定明[2] 黄哲学[1,2] 菲律普弗尼尔-维格[2]
[1](人工智能与数字经济广东省实验室(深圳) 广东深圳 518107)
[2](深圳大学计算机与软件学院 广东深圳 518060)
(wubo2022@email.szu.edu.cn)

Edge Hashing Distributed Sampling Algorithm for Triangle Counting in Large-scale Dynamic Graph Stream

He Yulin[1,2], Wu Bo[2], Wu Dingming[2], Huang Zhexue[1,2], and Philippe Fournier-Viger[2]
[1](*Guangdong Laboratory of Artificial Intelligence and Digital Economy* (*Shenzhen*), *Shenzhen*, *Guangdong* 518107)
[2](*College of Computer Science and Software Engineering*, *Shenzhen University*, *Shenzhen*, *Guangdong* 518060)

Abstract Triangle counting is a classical problem in large-scale graph analysis and widely applied in areas such as social network analysis, anomaly detection, and graph mining. In dynamic graph data streams, the topological structure of graph is constantly changing, involving edge insertions and deletions. Recent researches mainly focus on static streaming graphs and the corresponding streaming graph sampling algorithms only deal with edge insertions. Only a few single-machine sampling algorithms with low accuracy can handle edge deletion operations. We propose the edge hashing assignment-based distributed sampling (EHADS) algorithm, which is a distributed streaming algorithm designed for estimating the number of triangles in dynamic graph data streams. EHADS algorithm can rapidly and accurately estimate the counts of global triangles and local triangles associated with each vertex in dynamic streaming graphs. EHADS algorithm processes the input graph stream only once and samples edges across multiple machines. Compared with state-of-the-art single-machine streaming algorithms, the experimental results reflect that EHADS algorithm has the following advantages: 1) with the same sample capacity, EHADS algorithm achieves smaller estimation errors with less runtime, reducing the average estimation error of global triangle count by 31.79% and local triangle count by 23.35%; 2) EHADS algorithm provides unbiased estimations of the triangle count in streaming graph, and mathematical proofs demonstrate that the above-mentioned unbiased estimation has smaller variance.

Key words triangle counting; dynamic graph stream; edge sampling; distributed streaming algorithm; edge hashing assignment

摘 要 三角形计数是大图分析的一个经典问题,近年的研究工作主要集中在针对静态流式图的三角形数量估计上,相关流式图抽样算法只能处理边的插入操作,无法处理边的删除操作;而现有的动态流式图抽样算法估计准确性又偏低.针对上述问题,提出了基于边哈希分配的分布式抽样(edge hashing assignment-based distributed sampling, EHADS)算法,它是一个用于估计动态流式图中三角形数量的分布式流算法,

收稿日期: 2024-03-15; **修回日期**: 2024-05-17
基金项目: 广东省自然科学基金项目(2314050006683); 深圳市基础研究重点项目(JCYJ20220818100205012); 深圳市基础研究项目(JCYJ20210324093609026)
This work was supported by the Natural Science Foundation of Guangdong Province (2314050006683), the Key Basic Research Foundation of Shenzhen (JCYJ20220818100205012), and the Basic Research Foundation of Shenzhen (JCYJ20210324093609026).
通信作者: 何玉林(yulinhe@gml.ac.cn)

可以快速准确地估计动态流式图中的全局三角形数量以及每个顶点的局部三角形数量. EHADS 算法只对输入的图流进行 1 次处理，并在多台机器上对边进行抽样. 与先进的单机流算法相比，EHADS 算法具有 2 点优势：1）在相同样本容量的情况下，EHADS 算法以更短的运行时间获得了更小的估计误差，估计全局三角形数量的误差平均降低了 31.79%，估计局部三角形数量的误差平均降低了 23.35%；2）EHADS 算法能够提供流式图中三角形数量的无偏估计，并且严格的数学证明显示该无偏估计具有更小的方差.

关键词 三角形计数；动态图流；边抽样；分布式流算法；边哈希分配

中图法分类号 TP391

在现实世界中，图被广泛应用于各个领域，因为它提供了一种有效的方式来表示复杂的关系和结构. 现如今，大量的数据经常以图的形式来进行建模和分析[1]. 这些数据包括社交媒体网络[2]（如 Facebook 和 Twitter）、合作与合著网络[3]、交通网络[4]，以及多种类型的生物网络[5]等. 绝大多数社交网络，如 Twitter，Facebook 以及 MSN 等，拥有庞大的用户群体，数量通常在数百万到数十亿之间[6]. 在大数据时代，研究和分析这些规模庞大的网络存在着严峻的内存需求挑战. 在此背景下，一种常见的应对策略是利用抽样技术来对这些网络进行分析，从而得到近似的分析结果.

三角形计数是网络科学领域的一个核心问题，它是一种计算复杂度高但很重要的图统计量学. 真实世界的网络中通常存在大量的三角形，而随机生成的网络很少具有这种特征. 三角形结构通常被认为是用来区分真实世界网络和随机生成网络的重要特征. 三角形数量的计算涉及到网络分析中的许多重要度量，包括单个顶点的聚类系数[7]、整个网络的聚类系数[8]、传递性比率[9]以及三角形连通性[10]等. 社会科学家借助三角形计数来深入理解网络结构和关系[11-12]. 在社交网络中，三角形计数被用来发现社区[13]、检测虚假账户[14]、确定用户在网络中的角色[15]等. 三角形计数还被应用于许多图数据挖掘和数据库应用中，例如 Web 垃圾邮件的检测[16-17]、揭示 Web 中隐藏的主题关系[18]、密集子图的挖掘[19]、数据库的查询优化[20]、链接[21-22]推荐、异常检测等[23].

目前已经存在许多精确的单机三角形计数[24-27]算法. 鉴于其高昂的计算时间和内存空间要求，精确的单机三角形计数算法并不适用于大型图的处理. 为了能够处理大型图，另外一种广泛采用的方法是将三角形计数算法扩展到分布式环境中去对大型图进行计数[28-34]. 然而，与流算法不同，这些分布式计数算法需要一次性获取图中所有边的信息. 因此，分布式计数并不适用于流式图三角形计数，因为流式图的边是随着时间的推移而不断接收的，无法一次性获取全部边数据.

现实世界中许多图更适合用流式图来表示，这些图会随着时间的推移不断地变化. 流式图抽样算法通过对输入的边流进行 1 次遍历来估计图流中的三角形数量. 在遍历的过程中，流算法能够在每条边到达时增量式地更新三角形数量估计值，并且只需要存储一部分图的边. 因此，流式图抽样算法非常适用于维护和更新动态图流中的近似三角形计数. 在实时处理不断变化的图数据时，流算法可以随着边的添加和删除而持续提供三角形数量的估算，而无需从头开始重新计算，从而实现了高效的动态图分析. 这种特性对于需要不断监测和调整网络特性的应用非常有用，例如社交网络或互联网流量分析.

在动态的图数据流中，图的拓扑结构会不断发生变化，包含边的添加和删除操作. 然而，目前已有的三角形计数流算法主要专注于处理仅包含边插入操作的图数据流[20,23,35-48]. 大多数流式图抽样算法只能处理边的插入操作，只有少数几个流式图抽样算法具备处理边删除操作的能力[49-55]. 这些具备边删除操作处理能力的流算法都被设计成在单一计算机上运行. 据我们所了解，目前尚未有其他流算法能够有效地利用多台机器协同工作，以实现在动态图数据流中对三角形数量进行更快速或更精确的估算.

在本文中，我们提出了一个基于 Master-Worker-Aggregator 架构的分布式抽样算法 EHADS（edge hashing assignment-based distributed sampling），用于估计动态图流中的全局三角形和局部三角形数量. 图 1 展示了 EHADS 算法的体系结构，其中包括 1 个主节点、多个工作节点以及 1 个聚合节点. 在图 1 所示的图流中，黑色的边表示边的插入操作，而红色的边表示边的删除操作，这些边操作按照时间顺序输入. 在图流的处理中，边操作从数据源端流向主节点，然后主节点通过特定的边哈希分配策略，决定每条边操作将由哪些工作节点进行处理和采样. 如图 1 所示，主节

点会为每条插入的边设置一个哈希值，此外，每条边删除操作都会更新所有工作节点的样本图. 主节点的边哈希分配策略对通信开销、工作节点负载以及图流中三角形采样概率等方面具有重要影响. EHADS算法中主节点的边哈希分配策略由2个关键部分构成，即哈希模型和分发方式. EHADS算法采用边映射的哈希模型，通过应用随机边哈希函数为每条插入边生成一个哈希值，从而将每条插入边等概率地映射到某个工作节点上. 分发方式包括单播、组播和广播3种不同的方式. 为了尽可能地保证较大的三角形采样概率，在EHADS算法中，主节点选择广播作为分发方式，即主节点将每条边操作广播至所有工作节点. 因此，每个工作节点都会对原始图流中的所有边操作进行处理，计算当前输入边在本地样本图中构成三角形的数量，随后将计数结果发送给聚合节点. 工作节点在处理每条边的过程中，会根据主节点设置的边哈希值对插入边进行采样，并利用本地内存来维护一个样本图. 仅当插入边的哈希值与当前工作节点的序号相等时，该工作节点才会对其进行采样. 值得注意的是，插入边只能被映射到一个工作节点，但也有可能不被映射到任何工作节点. 如图1所示，图流中的插入边都被映射到了对应的工作节点. 这些工作节点依据本地存储的样本图来计算三角形的数量. 聚合节点负责汇总来自所有工作节点的三角形计数，并进行相应的处理，以获得对整个图流中全局三角形和局部三角形数量的估计.

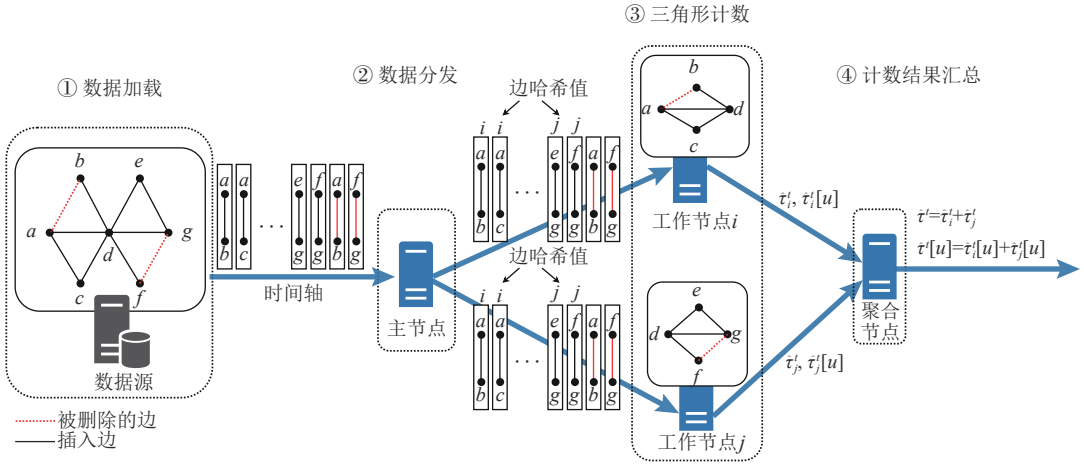


Fig. 1 The framework of EHADS algorithm

图1 EHADS算法框架

为了利用多台机器来获得更精确的三角形数量估计值，一种简单而有效的方式是直接并行化现有的单机流抽样算法. 具体而言，可以将图数据广播至所有机器，然后在每台机器上独立运行单机流抽样算法. 由于每台机器都能够提供三角形数量的无偏估计，通过对这些机器提供的估计结果取平均值，就可以得到一个更精确的三角形数量估计值. 这种方式能够减小估计结果的方差，从而提高估计的准确性. 现有的单机流抽样算法的估计误差受到共享边三角形对的协方差的显著影响. 尽管通过简单并行化单机流抽样算法可以在一定程度上提高估计准确性，但却并不能有效地减小由于共享边的三角形对引起的协方差. 本文提出的EHADS算法利用多台机器来采样动态流式图中的三角形，通过边哈希分配策略合理地规划了多个工作节点的样本边集. EHADS算法通过合理地利用多个工作节点样本边集之间的依赖关系，显著减小了共享边的三角形对引起的协方差.

我们的理论和实验分析表明，与最先进的流算法相比，EHADS算法具有2点优势：1)在相同样本容量的情况下，EHADS算法以更短的运行时间获得了更小的估计误差，估计全局三角形数量的误差平均降低了31.79%，估计局部三角形数量的误差平均降低了23.35%；2)EHADS算法能够提供流式图中三角形数量的无偏估计，并且严格的数学证明显示该无偏估计具有更小的方差.

1 相关工作

1.1 精确的三角形计数

目前已经存在许多精确的单机三角形计数[24]算法. 一个简单而精确的三角形计数算法涉及遍历整

个图的所有顶点，并在每个顶点处计算其邻居之间的边数，以确定存在多少个三角形. 该算法的时间复杂度可以用一个简单的上界来描述，即 $O\left(d_{\max}^2 n\right)$，其中 $d_{\max}$ 是图 G 中顶点最大的度，而 n 是图中的顶点数量. 另一个简单的计数算法是遍历整个图的所有边 (u, v)，然后计算顶点 u 和 v 的公共邻居[35]. 这个算法可以通过引入哈希技术来改进，从而使其时间复杂度的上界为 $O(m^{3/2})$，其中 m 是图中的边数. 目前，最快的单机三角形计数算法是基于矩阵乘法的计数算法. 给定图的邻接矩阵，可以在 $O(n^{\omega}) \subset O(n^{2.376})$ 的时间复杂度内高效地解决三角形计数问题[25-26]. Alon 等人[27]提出了一个时间复杂度为 $O(m^{2\omega/(\omega+1)}) \subset O(m^{1.41})$ 的三角形计数算法. 然而，这些高效的三角形计数算法均依赖于图的邻接矩阵，空间复杂度较高，这意味着在处理大型图时需要分配大量的内存来存储图的邻接矩阵. 因此，鉴于其较高的计算时间复杂度和内存空间复杂度，精确的单机三角形计数算法并不适用于处理大型图.

为了能够处理大型图，另一种广泛采用的方法是将三角形计数算法扩展到分布式环境中去进行计数. Cohen 等人[28]首次提出了一种适用于大型图中三角形计数的 MapReduce 算法. 这些分布式计数算法充分利用多台计算机的计算和存储资源，以便提高三角形计数算法的速度和可扩展性. 通过将计算任务分散到多台机器上，它们能够更快地处理大规模图数据，并且适应不断增长的计算需求. 然而，与流算法不同，这些分布式计数算法需要一次性获取图中所有边的信息. 因此，它们并不适用于流式图三角形计数，因为流式图的边是随着时间的推移而不断被接收，无法一次性获取全部边数据.

1.2 静态流式图中的近似三角形计数

现有的大多数针对流式图三角形计数的抽样算法只能处理静态流式图，这种类型的流式图如图 2(a)所示. 静态流式图以边为核心，采用边集合的方式来表示整个图结构，通常以边列表的形式进行存储. 静态流式图中只能包含边的插入操作，很多大型图可以自然地转换成这种类型的流式图. Bar-Yossef 等人[20]提出了第一个用于估计图流中三角形数量的流算法. Tsourakakis 等人[35]提出了一种使用稀疏化方法的流算法 Doulion，该算法以概率 p 独立采样每条边，然后利用样本图以及每个三角形被检测到的概率来估计图流中的全局三角形的数量. 一旦知道每个三角形被检测到(即被采样)的概率，就可以轻松获得对三角形数量的无偏估计. 从 Doulion 算法的方差分析中可以看出，这种无偏估计的方差大致与每个三角形被检测到的概率成反比. Pagh 等人[36]提出了一种基于着色的抽样方案，该算法通过随机地为图中的每个顶点涂上颜色并仅采样单色边，有效地将每个三角形被算法发现的概率从 p^3 提高到 p^2. Ahmed 等人[37]提出了一个通用的流式图抽样框架，可用于估计多种图属性，包括三角形数量等. Lim 等人[38]提出了一个用于估计流式图中局部三角形数量的单次遍历流算法 MASCOT. 在 MASCOT 中，每个三角形被采样的概率仅与该三角形中先到达的 2 条边有关. 然而，

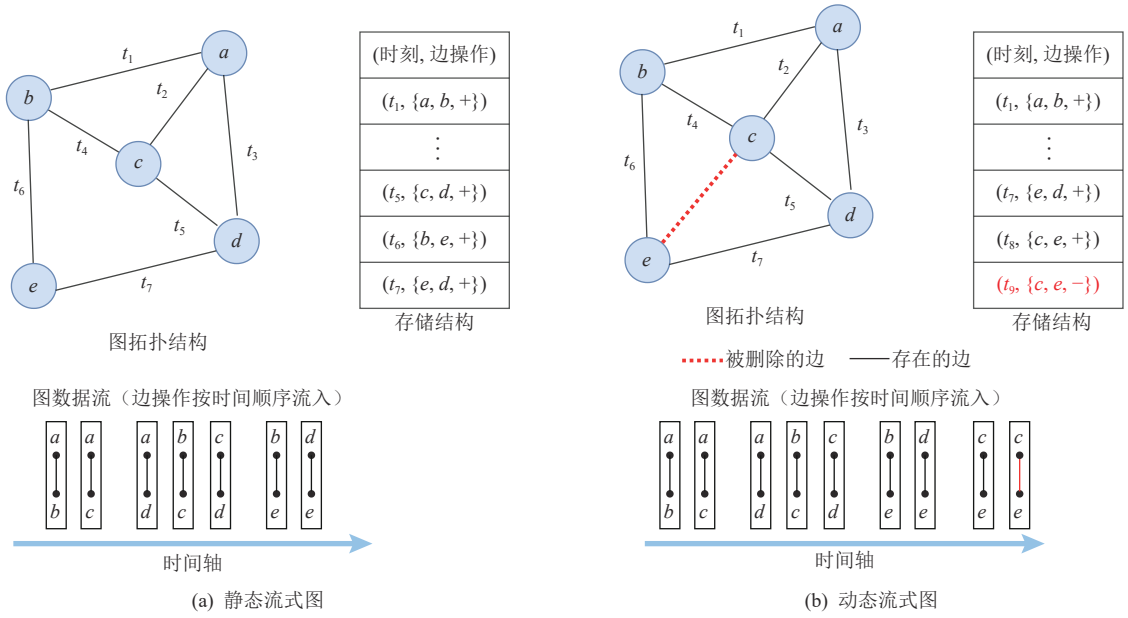


Fig. 2 Two different types of streaming graph

图 2 2 种不同类型的流式图

当样本容量未达到上限时，MASCOT 会丢弃边. Stefani 等人[52]提出了 Triest 算法，其中，Triest-IMPR 算法基于蓄水池抽样技术[56]. Triest-IMPR 算法充分利用了给定的样本容量，在样本容量未满时，会将每条到达的边添加至样本，有效地提高了三角形被抽样的概率，进而提高了无偏估计的准确性. Ahmed 等人[39]提出了一种新的加权边抽样算法 GPS. 然而，GPS 需要更多的内存来存储样本边的权值，并且处理这些权值的操作比较耗时. Shin 等人[40]提出了第一个用于估计图流中三角形数量的分布式流算法 Tri-Fly. Wang 等人[41]提出了适用于多核机器或机器集群的并行流算法 REPT. REPT 算法显著降低了采样的三角形之间的协方差，在某些情况下甚至完全消除了协方差. Shin 等人[42]在 Tri-Fly 算法的基础上开发了 CoCos 算法，显著降低了无偏估计的误差. 在 CoCos 算法中，图流中的每条边最多存储在 2 个工作节点中，而每个三角形只会被 1 个工作节点计数，这有效地减少了计算和存储资源的使用. 然而，CoCos 算法无法保证工作负载平衡. Yang 等人[43]将最先进的单机流算法 Triest 扩展为 4 种分布式流算法，这些算法采用多个主节点和分层分组的聚合器进行三角形计数，有效地提高了三角形计数的效率. Gou 等人[44]提出的 SWTC 算法基于滑动窗口模型能够高效地估计滑动窗口内部图流的三角形数量.

1.3 动态流式图中的近似三角形计数

在动态流式图中，图的拓扑结构会随着时间不断演化，图流中包含边的添加和删除操作. 如图 2(b)所示，动态流式图呈现了一个不断变化的网络. Kutzkov 等人[49]提出了第一个用于估计动态流式图中三角形数量的算法. 他们将前面提到的顶点着色抽样方案应用于动态流式图，然后使用该抽样方案，以并行的方式创建多个稀疏化的样本图. 紧接着该算法根据这些样本图来估计输入图流中形成三角形的二路径(2 个相连接的边的序列)的比例. 最后，通过将这个估计的比例与输入图流中总二路径数量的估计值相乘，算法就可以得出全局三角形数量的估计值. 然而，该算法并不适用于实时应用程序，因为它不能持续地更新三角形数量估计值. 相反，只能在输入图流的末尾一次性估计全局三角形的数量. 此外，在最坏的情况下，该算法所需的内存资源会超过存储整个图所需的内存. Han 等人[51]提出了 ESD 算法，ESD 算法在处理图流中的每条边操作时，能够持续地提供全局三角形数量的估计值. 然而，在处理每个到达的边操作时，ESD 算法需要访问当前时刻的整个图，也就是说，它需要维护当前输入图的状态. Stefani 等人[52]提出的 Triest-FD 算法能够用于估计动态流式图中全局和局部三角形的数量，该算法基于随机配对(random pairing, RP)抽样技术[57]. Triest-FD 算法利用 RP 抽样技术来维护特定大小的均匀边样本，然后根据样本图中的三角形数量以及每个三角形被采样的概率，实现了对三角形数量的无偏估计. Shin 等人[53]提出的 ThinkD 算法在处理每个新到达的边操作时，首先用该边操作来更新其维护的三角形数量估计值，接着再利用该边操作来更新其维护的样本图. ThinkD 算法有效地增加了每个添加或删除的三角形被采样的概率，从而降低了无偏估计的方差. ThinkD 算法有 2 个版本，分别是 ThinkDfast 和 ThinkDacc. 其中，ThinkDfast 算法具有更快的运行速度，而 ThinkDacc 算法则提供更准确的无偏估计. 如果图流中的边序列表现出显著的时间局部性，则 Lee 等人[54]提出的 WRS 算法将更为适用. WRS 算法将给定的样本空间划分为等候室和蓄水池，图流中新到达的边会直接被采样并存放在等候室中. 在 WRS 算法中，图流中的三角形被分为 3 种类型. WRS 算法的准确性取决于图流中各种三角形类型的分布，类型 1 和类型 2 的三角形越多，WRS 算法提供的估计值就会越准确. Yu 等人[55]基于三角形的第一条到达边提出了 RFES 算法. 在该算法中，每条样本边都会维护 2 个邻居列表. 不论当前输入边是否被采样，该输入边都会更新水库中样本边的邻居列表. 这一策略使得图流中每个三角形被采样的概率仅与该三角形的第 1 条到达边相关. 然而，值得注意的是，RFES 算法中样本边邻居列表所占用的空间通常远远大于所设定的样本容量.

2 符号与问题定义

本节介绍了本文所采用的符号体系，并明确定义了动态图流中分布式三角形计数所涉及的关键问题. 表 1 详细列出了在本文中所使用的符号及其含义.

2.1 符号与概念

定义动态无向图流Π为一个边操作序列$(e^1, e^2, \cdots, e^t)$，其中e^t表示第t个到达的边操作. 图$\mathcal{G}^t$是通过图流Π中前t个到达的边操作逐步生成的，初始为空图，其中$\mathcal{V}^t$和$\mathcal{E}^t$分别表示图$\mathcal{G}^t$中顶点和边的集合. 对于每条边$\{u,v\}\in\mathcal{E}^t$，它连接了 2 个不同的节点，即$u\neq v\in\mathcal{V}^t$. 符号$e^t=\{u,v,\delta\}$表示在时刻t添加或删除边$\{u,v\}$，其中$\delta\in\{+,-\}$表示图$\mathcal{G}$中的变化. 本文约定只能对图$\mathcal{G}$执行 2 种操作：添加新边和删除已存在的边.

Table 1 Symbol Definitions Employed in Our Paper

表 1 本文所采用的符号定义

符号	定义
Π	动态图流（按时间输入的边操作序列）
e^t	时刻t到达的边操作
$\mathcal{G}^t=(\mathcal{V}^t,\mathcal{E}^t)$	在时刻t的原始图
$\mathcal{S}_i^t$	在时刻t第i个工作节点维护的样本集
$\mathcal{N}_i^t[u]$	样本集$\mathcal{S}_i^t$中顶点u的邻居集合
$\{u,v\}$	顶点u和顶点v之间的边
$\{u,v,w\}$	由顶点u,v,w形成的三角形
$\mathcal{T}^t$	图$\mathcal{G}^t$中全部三角形的集合
$\mathcal{T}^t[u]$	与顶点u相关的局部三角形集合
$\hat{\tau}$	图中全局三角形数量的估计值
$\hat{\tau}[u]$	图中与顶点u相关的局部三角形数量的估计值
m	哈希函数的映射范围
$p=1/m$	边采样概率
$h(\cdot)$	EHADS 算法的边哈希函数
$h_j(\cdot)$	第j组工作节点的边哈希函数
$\mathcal{A}^t$	在时刻t之前图流中新增的所有三角形集合
$\mathcal{D}^t$	在时刻t之前图流中被删除的所有三角形集合
$\mathcal{M},\mathcal{W},\mathcal{C}$	主节点、工作节点、聚合节点的集合

在图$\mathcal{G}^t$中，对于 3 个顶点u，v和w，如果满足任意 2 个顶点之间都存在边（即$e_{uv},e_{uw},e_{vw}\in\mathcal{E}^t$），那么称$\{u,v,w\}$形成一个三角形. 本文使用符号$\mathcal{T}^t$来表示图$\mathcal{G}^t$的全局三角形的集合，即图中的所有三角形. 此外，符号$\mathcal{T}^t[u]$表示每个顶点$u\in\mathcal{V}^t$的局部三角形的集合，即包含顶点u的所有三角形. 如图 3 所示，顶点a具有 2 个局部三角形，分别为$\{a,b,d\}$和$\{a,c,d\}$. 整个示例图中总共有 6 个不同的三角形，因此整张图的全局三角形数量为 6. 本文采用符号$\mathcal{A}^t$和$\mathcal{D}^t$来描述动态图流中被添加和删除的三角形集合，其具体含义详见定义 1 和定义 2.

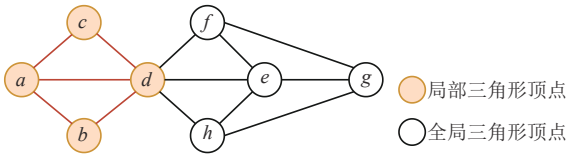


Fig. 3 Examples of global triangles and local triangles

图 3 全局三角形与局部三角形示例

定义 1. 添加的三角形集合. 定义$\mathcal{A}^t$为在时刻t或之前已经添加到图$\mathcal{G}$中的三角形的集合. 具体而言，$\mathcal{A}^t$表示以下形式的三角形集合：

$$\mathcal{A}^t:=\{(\{u,v,w\},s)\mid 1\leqslant s\leqslant t,\{u,v,w\}\notin\mathcal{T}^{s-1},\{u,v,w\}\in\mathcal{T}^s\}. \tag{1}$$

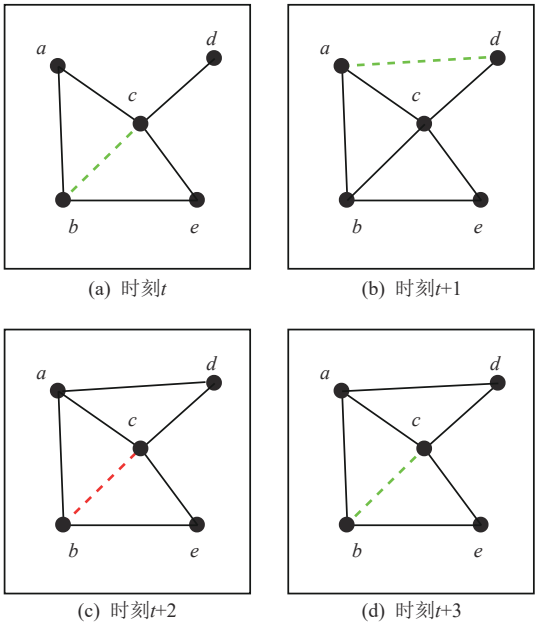
在定义 1 中，引入了时间标识符s，以区分具有相同顶点组成但在不同时间点添加到图中的三角形.

定义 2. 删除的三角形集合. 定义$\mathcal{D}^t$为在时刻t或之前从图$\mathcal{G}$中删除的三角形的集合. 具体而言，$\mathcal{D}^t$表示的三角形集合形式为：

$$\mathcal{D}^t:=\{(\{u,v,w\},s)\mid 1\leqslant s\leqslant t,\{u,v,w\}\in\mathcal{T}^{s-1},\{u,v,w\}\notin\mathcal{T}^s\}. \tag{2}$$

在定义 2 中，引入了时间标识符s，以区分具有相同顶点组成但在不同时间点被删除的三角形.

同理，当涉及图中每个顶点$u\in\mathcal{V}^t$时，采用$\mathcal{A}^t[u]\subset\mathcal{A}^t$和$\mathcal{D}^t[u]\subset\mathcal{D}^t$来表示包含顶点$u$的已添加和已删除的三角形集合. 需要注意的是，具有相同顶点组成的三角形可以被添加多次，也可以被删除多次. 在图 4 中，黑色实线表示当前时刻之前就已经存在的边，绿色虚线表示当前时刻添加的边，红色虚线表示当前时刻删除的边. 如图 4 所示，在时刻t，一条边$\{b,c\}$被添加到图中，从而引发了图流中 2 个三角形的添加，这 2 个三角形分别为$\{a,b,c\}$和$\{b,c,e\}$. 在时刻t+2，图中的边$\{b,c\}$被删除，导致了三角形$\{a,b,c\}$和$\{b,c,e\}$的移除. 在时刻t+3，边$\{b,c\}$被重新添加到图中，进而再次引发了三角形$\{a,b,c\}$和$\{b,c,e\}$的添加. 根据添加三角形的定义，在时刻t添加的三角形集合为$\mathcal{A}^t-\mathcal{A}^{t-1}$. 同理，根据删除三角形的定义，在时刻$t$删除的三角形集合为$\mathcal{D}^t-\mathcal{D}^{t-1}$. 在图 4 中，集合$\mathcal{A}^{t+3}-\mathcal{A}^{t-1}$包含的元素为：$(\{a,b,c\},t)$，$(\{b,c,e\},t)$，$(\{a,c,d\},t+1)$，$(\{a,b,c\},t+3)$，$(\{b,c,e\},t+3)$. 同理，集合$\mathcal{D}^{t+3}-\mathcal{D}^{t-1}$包含的元



(a) 时刻t (b) 时刻t+1

(c) 时刻t+2 (d) 时刻t+3

Fig. 4 Examples of changes in the number of triangles in dynamic graph

图 4 动态图中的三角形数量变化示例

素为：({*a*, *b*, *c*}, *t*+2), ({*b*, *c*, *e*}, *t*+2).

2.2 问题定义

本文工作研究了通过多台机器来估计动态图流(即一系列边操作)中全局和局部三角形数量的问题.具体来说，考虑了3点现实条件：

1)无先验知识.算法在处理图流之前没有任何关于输入图流的先验知识，如顶点和边的数量信息.

2)无共享环境.每台机器上的存储数据是相互独立的，即存储在一台机器上的数据无法被其他机器访问或共享.

3)单次遍历.每台机器遵循单次遍历的原则，按照时间顺序逐个处理图流中的边操作.每台机器只能访问当前正在处理的边操作以及本地存储的样本图中的边，无法访问已处理过的边操作.

基于以上3个条件，定义了动态图流中全局和局部三角形数量的分布式估计问题：

1)前提条件.有一个动态图流$(e^1,e^2,\cdots)$和$|\mathcal{W}|$个工作节点.

2)计算任务.对于每个时刻$t\in\{t_1,t_2,\cdots\}$，我们的任务是计算出图流中全局三角形数量$|\mathcal{T}^t|$的估计值$\hat{\tau}$，以及每个顶点$u\in\mathcal{V}^t$的局部三角形数量$|\mathcal{T}^t[u]|$的估计值$\hat{\tau}[u]$.

3)优化目标.最终目标是最小化估计量$\hat{\tau}$和$\hat{\tau}[u]$的偏差和方差，以确保算法给出的估计值尽可能接近真实数值.

3 本文提出的算法 EHADS

在本节中，详细介绍了EHADS算法，该算法基于Master-Worker-Aggregator架构，用于估计动态图流中的全局三角形和局部三角形数量.首先详细介绍了EHADS算法的2种情况：EHADS($m\geqslant|\mathcal{W}|$)和EHADS-g($m<|\mathcal{W}|$).然后对EHADS算法的复杂性进行了理论分析.

3.1 EHADS

EHADS算法的伪代码如算法1所示.在阐释伪代码之前，首先对其中采用的符号进行了定义.接着，详细阐述了主节点、工作节点和聚合节点各自的处理过程.最后，探讨了延迟聚合的机制.

算法1： EHADS算法($m\geqslant|\mathcal{W}|$).

输入：动态图流$\Pi=\left(e^1,e^2,\cdots,e^t\right)$，哈希函数的映射范围$m$；

输出：全局三角形数量的估计值$\hat{\tau}$，每个顶点u的局部三角形数量的估计值$\hat{\tau}[u]$.

主节点：

① for each operation $e^t=\{u,v,\delta\}$ from the graph stream Π do
② generate $h(e^t)$ randomly from $\{1,2,\cdots,m\}$;
③ send $(e^t,h(e^t))$ to all workers;
④ end for

工作节点(每个工作节点$i\in\mathcal{W}$)：

⑤ $\mathcal{S}_i\leftarrow\varnothing$;
⑥ for each element $(e^t,h(e^t))$ from the master do
⑦ *Update*(e^t);
⑧ if $\delta=+$ and $h(e^t)=i$ then *Insert*($\{u,v\}$);
⑨ else *Delete*($\{u,v\}$);
⑩ end if
⑪ end for
⑫ function *Update*(e^t):
⑬ $sum\leftarrow 0$;
⑭ for each common neighbor $w\in\mathcal{N}_i[u]\cap\mathcal{N}_i[v]$ do
⑮ if $\delta=+$ then
⑯ send $(w,1/(q[uvw]))$ to aggregator;
⑰ $sum\leftarrow sum+1/(q[uvw])$;
⑱ else
⑲ send $(w,-1/(q[uvw]))$ to aggregator;
⑳ $sum\leftarrow sum-1/(q[uvw])$;
㉑ end if
㉒ end for
㉓ send $(\#,sum)$, (u,sum) and (v,sum) to aggregator;
㉔ function *Insert*($\{u,v\}$):
㉕ $\mathcal{S}_i\leftarrow\mathcal{S}_i\cup\{\{u,v\}\}$;
㉖ function *Delete*($\{u,v\}$):
㉗ if $\{u,v\}\in\mathcal{S}_i$ then $\mathcal{S}_i\leftarrow\mathcal{S}_i-\{\{u,v\}\}$;
㉘end if

聚合节点：

㉙ $\hat{\tau}\leftarrow 0$;
㉚ $\hat{\tau}[u]\leftarrow 0,u\in\mathcal{V}^t$;
㉛ for each element (u,Δ) from the workers do
㉜ if $u=\#$ then $\hat{\tau}\leftarrow\hat{\tau}+\Delta$;
㉝ else $\hat{\tau}[u]\leftarrow\hat{\tau}[u]+\Delta$;
㉞ end if
㉟ end for

本节采用符号$\mathcal{M}$，$\mathcal{W}$和$\mathcal{C}$分别表示主节点、工作节点和聚合节点的集合.每个工作节点都能够存储一个样本图，符号$\mathcal{S}_i$表示工作节点i当前所保存的样本边集合.定义$\mathcal{G}_i=(\mathcal{V}_i,\mathcal{E}_i)$为由工作节点$i$所存储的边组成的图.对于每个顶点$u\in\mathcal{V}_i$，$\mathcal{N}_i[u]$表示$\mathcal{G}_i$中$u$的

邻居顶点集合. 最后，$\hat{\tau}$表示全局三角形数量的估计值，$\hat{\tau}[u]$表示顶点u的局部三角形数量的估计值.

针对每个来自源节点的边操作，主节点利用随机边哈希函数$h(\cdot)$将其均匀、随机地映射到整数集合$\{1,2,\cdots,m\}$，即$P(h(e^t)=i)=1/m$，其中$i\in\{1,2,\cdots,|\mathcal{W}|\}$，并且$m\geqslant|\mathcal{W}|$. 主节点会将边操作和对应的边哈希值$(e^t,h(e^t))$发送至所有工作节点（行①~④）.

每个工作节点首先将其本地存储的边样本集$\mathcal{S}_i$初始化为空. 每当工作节点接收到一条边操作$(e^t=\{u,v,\delta\},h(e^t))$时，工作节点会调用函数 *Update* 来更新聚合节点维护的估计量. 更新估计量的过程涉及计算由边$\{u,v\}$和当前样本图中的2条边所构成的三角形数量，并将计数结果发送至聚合节点. 随后，如果e^t是边的插入操作（$\delta=+$）并且其边哈希值为当前工作节点的序号（$h(e^t)=i$），工作节点则会通过调用 *Insert* 函数对边$\{u,v\}$进行采样，即将边$\{u,v\}$插入样本图. 而若e^t是边的删除操作（$\delta=-$），工作节点则通过调用 *Delete* 函数来从样本图中移除边$\{u,v\}$. 无论边删除操作的哈希值是否与当前工作节点的序号匹配，工作节点都将执行 *Delete* 函数从样本图中移除边$\{u,v\}$（行⑤~⑪）.

在 *Update* 函数的实现中，每个工作节点需要计算图$\mathcal{G}_i$（由$\mathcal{S}_i$中的样本边组成的图）中顶点u和v的公共邻居数量. EHADS 算法依赖于这样一个事实：每个公共邻居w意味着三角形$\{u,v,w\}$的存在，即工作节点成功采样到了三角形$\{u,v,w\}$. 对于每个发现到的公共邻居w，工作节点需要将相应的计数信息发送至聚合节点. 这个计数信息用来更新全局三角形数量（使用#符号来唯一标识）以及顶点u，v和w局部三角形数量的估计值. 当e^t是边的插入操作时，这表明该工作节点采样到了添加的三角形$(\{u,v,w\},t)\in\mathcal{A}^t$. 此时，工作节点需要将相关的计数信息发送至聚合节点，以使得全局三角形数量和相应的局部三角形数量的估计值增加，增量为$1/q[uvw]$. 如果e^t是边删除操作，这表示工作节点采样到了删除的三角形$(\{u,v,w\},t)\in\mathcal{D}^t$. 在这种情况下，工作节点需要发送相应的计数信息至聚合节点，以减少全局三角形数量和相应的局部三角形数量的估计值，减少的量为$1/q[uvw]$. 这种更新机制确保了动态图流中每个三角形所对应的随机变量的数学期望恰好为1或者–1. 其中，$q[uvw]$的值为：

$$q[uvw]=\frac{|\mathcal{W}|}{m^2}. \tag{3}$$

针对每个被添加或被删除的三角形$\{u,v,w\}$，$q[uvw]$是 EHADS 算法采样到三角形$\{u,v,w\}$的概率. 当样本$\mathcal{S}_i$中存在三角形$\{u,v,w\}$的前2条边$\{u,w\}$和$\{v,w\}$（即图流Π中先出现的2条边）时，工作节点i就能够成功采样到三角形$\{u,v,w\}$. 边$\{u,w\}$和$\{v,w\}$同时存在于样本$\mathcal{S}_i$的概率为$1/m^2$，即$P(h(\{u,w\})=i\wedge h(\{v,w\})=i)=1/m^2$. 由于每条边操作都会被主节点广播至所有工作节点，因此三角形被采样的概率不受第3条到达边的影响. 只要边$\{u,w\}$和$\{v,w\}$被哈希函数$h(\cdot)$映射到同一个工作节点，EHADS 算法就能够成功采样到三角形$\{u,v,w\}$. 因此，EHADS 采样到$\{u,v,w\}$的概率为$P(h(\{u,w\})=h(\{v,w\}))=\binom{|\mathcal{W}|}{1}\times\frac{1}{m^2}=\frac{|\mathcal{W}|}{m^2}$. 图流中的每个边删除操作都会更新每个工作节点本地存储的样本图. 因此，具有相同顶点组成但在不同时间点添加或删除的三角形被采样的概率都是相同的（行⑫~㉓）.

聚合节点会收集汇总所有工作节点发送的三角形计数结果，并进行相应的处理. 在伪代码中，#符号用来标识全局三角形数量计数. 为了得到整个图流中全局三角形数量以及每个顶点局部三角形数量的无偏估计，聚合节点需要对工作节点发送的计数结果进行求和（行㉙~㉟）.

在 EHADS 算法中，当每个工作节点采样到三角形$\{u,v,w\}$时，会立刻将相应的计数结果发送至聚合节点，以更新全局三角形数量和顶点u，v，w的局部三角形数量的估计值. 然而，在用户未查询图流中三角形数量时，可以通过引入一种称为惰性聚合的策略来最小化工作节点与聚合节点之间的网络通信. 这意味着，在用户发出三角形数量查询之前，工作节点可以在本地部分聚合计数结果，避免立即将计数结果发送至聚合节点. 惰性聚合允许工作节点暂时保留和本地汇总三角形计数信息. 只有当用户确实需要查询图流中的全局三角形和局部三角形数量时，工作节点才将预先聚合的本地结果发送至聚合节点进行最终的全局汇总和计算. 这种策略减少了在用户查询前不必要的网络通信，从而提高了算法的效率和资源利用率.

接下来，将证明 EHADS 算法提供了图流中全局三角形数量和局部三角形数量的无偏估计. 估计量的偏差和方差是决定估计误差的2个关键因素. 在论证 EHADS 算法的无偏性和方差之前，需要介绍一个关键的引理，即引理1，它描述了当前图流中三角形的数量.

引理 1. 当前图中的三角形数量. 在图流中，当前时刻图中的三角形数量可表示为已添加的三角形数量减去已删除的三角形数量. 形式上可表述为：

$$\left|\mathcal{T}^t\right|=\left|\mathcal{A}^t\right|-\left|\mathcal{D}^t\right|,\ \forall t\geqslant 1, \tag{4}$$

$$\left|\mathcal{T}^t[u]\right| = \left|\mathcal{A}^t[u]\right| - \left|\mathcal{D}^t[u]\right|, \forall t \geqslant 1, \forall u \in \mathcal{V}^t. \tag{5}$$

引理 1 提供了关于当前图中三角形数量的重要性质, 为定理 1 的证明提供了理论基础. 引理 1 清晰地表明了三角形数量与已添加和已删除的三角形之间的关系. 定理 1 证明了 EHADS 算法的无偏性, 即 EHADS 算法能够提供动态图流中三角形数量的无偏估计.

定理 1. EHADS 的无偏性. 对于任意时间 t, 当 $m \geqslant |\mathcal{W}|$时, EHADS 算法均能够提供$\mathcal{G}^t$中全局三角形数量和局部三角形数量的无偏估计. 定义$\hat{\tau}^t$和$\hat{\tau}^t[u]$分别表示算法处理了e^t后给出的$\hat{\tau}$和$\hat{\tau}[u]$. 基于此, 有结论为:

$$\mathbb{E}\left[\hat{\tau}^t\right] = \left|\mathcal{T}^t\right|, \forall t \geqslant 1, \tag{6}$$

$$\mathbb{E}\left[\hat{\tau}^t[u]\right] = \left|\mathcal{T}^t[u]\right|, \forall t \geqslant 1. \tag{7}$$

证明. 定义a^s_{uvw}表示工作节点采样到三角形$(\{u,v,w\},s) \in \mathcal{A}^t$而导致$\hat{\tau}$, $\hat{\tau}[u]$, $\hat{\tau}[v]$和$\hat{\tau}[w]$发生变化的增量. 在不失一般性的情况下, 假设$e^s = \{u,v,+\}$, 其中 $s \leqslant t$. 根据这些定义和假设, 有结论成立:

$$a^s_{uvw} = \begin{cases} \dfrac{1}{q[uvw]}, & h(\{u,w\}) = h(\{v,w\}) \leqslant |\mathcal{W}|; \\ 0, & \text{其他}. \end{cases} \tag{8}$$

结合式(8)和数学期望的性质时, 可以得出结论:

$$\mathbb{E}\left[a^s_{uvw}\right] = 1. \tag{9}$$

同理, 定义d^s_{uvw}表示工作节点采样到三角形$(\{u,v,w\},s) \in \mathcal{D}^t$而导致$\hat{\tau}$, $\hat{\tau}[u]$, $\hat{\tau}[v]$和$\hat{\tau}[w]$发生变化的减量. 假设$e^s = \{u,v,-\}$, 那么结论成立:

$$d^s_{uvw} = \begin{cases} \dfrac{-1}{q[uvw]}, & h(\{u,w\}) = h(\{v,w\}) \leqslant |\mathcal{W}|; \\ 0, & \text{其他}. \end{cases} \tag{10}$$

结合式(10)和数学期望的性质时, 可以得出结论:

$$\mathbb{E}\left[d^s_{uvw}\right] = -1. \tag{11}$$

根据定义 1 和定义 2, EHADS 算法维护的全局三角形数量估计值满足等式:

$$\hat{\tau}^t = \sum_{(\{uvw\},s)\in\mathcal{A}^t} a^s_{uvw} + \sum_{(\{uvw\},s)\in\mathcal{D}^t} d^s_{uvw}. \tag{12}$$

根据引理 1、式(9)(11)(12), 以及数学期望的线性性质, 可以推导出:

$$\mathbb{E}\left[\hat{\tau}^t\right] = \sum_{(\{uvw\},s)\in\mathcal{A}^t} \mathbb{E}\left[a^s_{uvw}\right] + \sum_{(\{uvw\},s)\in\mathcal{D}^t} \mathbb{E}\left[d^s_{uvw}\right] = \sum_{(\{uvw\},s)\in\mathcal{A}^t} 1 + \sum_{(\{uvw\},s)\in\mathcal{D}^t} -1 = \left|\mathcal{A}^t\right| - \left|\mathcal{D}^t\right| = \left|\mathcal{T}^t\right|. \tag{13}$$

因此, 根据上述推导可以得出式(6)成立. 同样, 对于每个顶点$u \in \mathcal{V}^t$, EHADS 算法维护的局部三角形数量估计值满足:

$$\hat{\tau}^t[u] = \sum_{(\{uvw\},s)\in\mathcal{A}^t[u]} a^s_{uvw} + \sum_{(\{uvw\},s)\in\mathcal{D}^t[u]} d^s_{uvw}. \tag{14}$$

同理, 根据引理 1、式(9)(11)(14), 以及数学期望的线性性质, 可以推导出:

$$\mathbb{E}\left[\hat{\tau}^t[u]\right] = \sum_{(\{uvw\},s)\in\mathcal{A}^t[u]} \mathbb{E}\left[a^s_{uvw}\right] + \sum_{(\{uvw\},s)\in\mathcal{D}^t[u]} \mathbb{E}\left[d^s_{uvw}\right] = \sum_{(\{uvw\},s)\in\mathcal{A}^t[u]} 1 + \sum_{(\{uvw\},s)\in\mathcal{D}^t[u]} -1 = \left|\mathcal{A}^t[u]\right| - \left|\mathcal{D}^t[u]\right| = \left|\mathcal{T}^t[u]\right|. \tag{15}$$

因此, 根据上述推导可以得出式(7)成立. 证毕.

接下来, 将推导并证明 EHADS 算法维护的$\hat{\tau}^t$和$\hat{\tau}^t[u]$的方差. 在深入分析 EHADS 算法的方差之前, 需要引入一个关键的定义, 即定义 3, 对当前图流中的三角形对类型明确定义. 该定义提供了三角形对的精确分类, 为后续方差分析提供了理论基础. 对于当前图流中每个添加或删除的三角形$(\{u,v,w\},s) \in \mathcal{A}^t \cup \mathcal{D}^t$, 引入指示函数$\mathbf{1}_{(\{u,v,w\},s)}$表示与该三角形第 1 个到达的边相关联的边插入操作, 同时使用$\mathbf{2}_{(\{u,v,w\},s)}$表示与第 2 个到达的边相关联的边插入操作.

定义 3. 三角形对的类型. 对于任意 2 个不同的三角形$\sigma \neq \omega \in \mathcal{A}^t \cup \mathcal{D}^t$, 其有序对的类型定义为:

$$Type_{(\sigma,\omega)} = \begin{cases} 1, & \text{if } \sigma \in \mathcal{A}^t \text{ and } \omega \in \mathcal{A}^t \text{ and } |\{\mathbf{1}_\sigma, \mathbf{2}_\sigma\} \cap \{\mathbf{1}_\omega, \mathbf{2}_\omega\}| = 1, \\ 2, & \text{if } \sigma \in \mathcal{D}^t \text{ and } \omega \in \mathcal{D}^t \text{ and } |\{\mathbf{1}_\sigma, \mathbf{2}_\sigma\} \cap \{\mathbf{1}_\omega, \mathbf{2}_\omega\}| = 1, \\ 3, & \text{if } \sigma \in \mathcal{A}^t \text{ and } \omega \in \mathcal{D}^t \text{ and } |\{\mathbf{1}_\sigma, \mathbf{2}_\sigma\} \cap \{\mathbf{1}_\omega, \mathbf{2}_\omega\}| = 1, \\ 4, & \text{if } \sigma \in \mathcal{D}^t \text{ and } \omega \in \mathcal{A}^t \text{ and } |\{\mathbf{1}_\sigma, \mathbf{2}_\sigma\} \cap \{\mathbf{1}_\omega, \mathbf{2}_\omega\}| = 1, \\ 5, & \text{if } \sigma \in \mathcal{A}^t \text{ and } \omega \in \mathcal{A}^t \text{ and } |\{\mathbf{1}_\sigma, \mathbf{2}_\sigma\} \cap \{\mathbf{1}_\omega, \mathbf{2}_\omega\}| = 2, \\ 6, & \text{if } \sigma \in \mathcal{D}^t \text{ and } \omega \in \mathcal{D}^t \text{ and } |\{\mathbf{1}_\sigma, \mathbf{2}_\sigma\} \cap \{\mathbf{1}_\omega, \mathbf{2}_\omega\}| = 2, \\ 7, & \text{if } \sigma \in \mathcal{A}^t \text{ and } \omega \in \mathcal{D}^t \text{ and } |\{\mathbf{1}_\sigma, \mathbf{2}_\sigma\} \cap \{\mathbf{1}_\omega, \mathbf{2}_\omega\}| = 2, \\ 8, & \text{if } \sigma \in \mathcal{D}^t \text{ and } \omega \in \mathcal{A}^t \text{ and } |\{\mathbf{1}_\sigma, \mathbf{2}_\sigma\} \cap \{\mathbf{1}_\omega, \mathbf{2}_\omega\}| = 2, \\ 9, & \text{其他 (i.e., } |\{\mathbf{1}_\sigma, \mathbf{2}_\sigma\} \cap \{\mathbf{1}_\omega, \mathbf{2}_\omega\}| = 0). \end{cases} \tag{16}$$

如图 5 所示, 动态图流中的三角形对共享边的情况可分为 8 种类型. 在图 5 中, 绿色边表示某个添加的三角形$\sigma \in \mathcal{A}^t$中第 3 个到达的边, 而红色边表示某个删除的三角形$\sigma \in \mathcal{D}^t$中第 3 个到达的边. 具体而言, 图 5 中的类型 1~4 的三角形对仅共享 1 条边, 这 1 条共享边所对应的边操作为$(t_3, \{u,v,+\})$. 而类型 5~8 的三角形对共享 2 条边, 这 2 条共享边所对应的边操作分别为$(t_1, \{u,w,+\})$和$(t_2, \{v,w,+\})$. 值得注意的是, 图 5 中只呈现了每种类型三角形对的一种情况, 实际上每种类型可能存在多种情形.

定理 2. EHADS 的方差. 对于任意时间 $t \geqslant 1$, 定义n^t_i为在$\mathcal{A}^t \cup \mathcal{D}^t$中类型$i$的三角形对的数量, 同时定

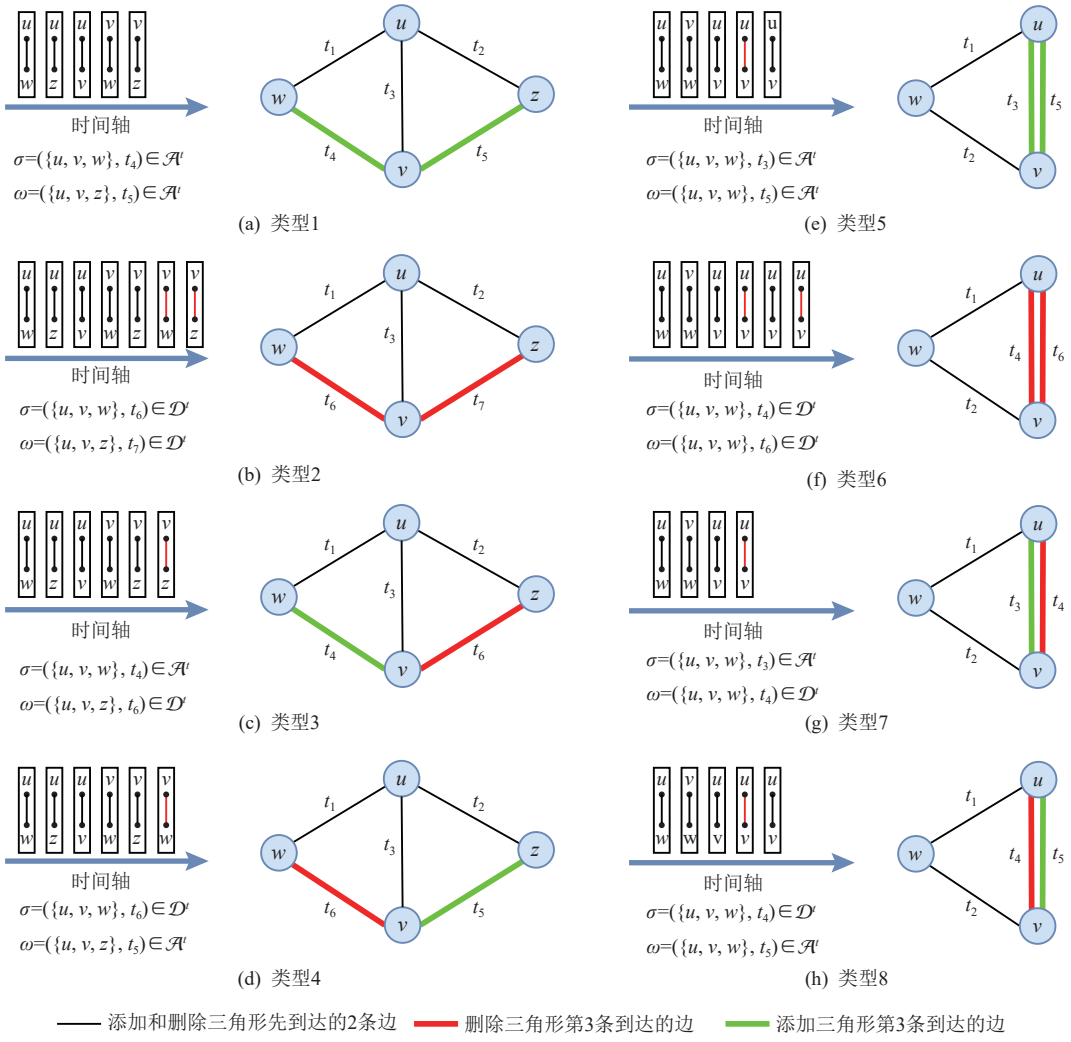


Fig. 5 Eight different types of triangle pair in dynamic graph stream

图 5 动态图流中 8 种不同类型的三角形对

义 $n_i^t[u]$ 为在 $\mathcal{A}^t[u]\cup\mathcal{D}^t[u]$ 中类型 i 的三角形对的数量. 那么, EHADS 算法所维护的 $\hat{\tau}^t$ 和 $\hat{\tau}^t[u]$ 的方差满足:

$$\begin{aligned}Var\left[\hat{\tau}^t\right]=&\left(\left|\mathcal{A}^t\right|+\left|\mathcal{D}^t\right|\right)\left(\frac{m^2}{|\mathcal{W}|}-1\right)+\\&\left(n_1^t+n_2^t-n_3^t-n_4^t\right)\left(\frac{m}{|\mathcal{W}|}-1\right)+\\&\left(n_5^t+n_6^t-n_7^t-n_8^t\right)\left(\frac{m^2}{|\mathcal{W}|}-1\right),\ \forall t\geqslant 1.\end{aligned}\tag{17}$$

$$\begin{aligned}Var\left[\hat{\tau}^t[u]\right]=&\left(\left|\mathcal{A}^t[u]\right|+\left|\mathcal{D}^t[u]\right|\right)\left(\frac{m^2}{|\mathcal{W}|}-1\right)+\\&\left(n_1^t[u]+n_2^t[u]-n_3^t[u]-n_4^t[u]\right)\left(\frac{m}{|\mathcal{W}|}-1\right)+\\&\left(n_5^t[u]+n_6^t[u]-n_7^t[u]-n_8^t[u]\right)\left(\frac{m^2}{|\mathcal{W}|}-1\right),\\&\forall t\geqslant 1,\ \forall u\in\mathcal{V}^t.\end{aligned}\tag{18}$$

证明. 对于每个三角形 $\sigma=(\{u,v,w\},s)\in\mathcal{A}^t\cup\mathcal{D}^t$, 定义 γ_σ 表示由于工作节点采样到 σ 而导致 $\hat{\tau}$, $\hat{\tau}[u]$、$\hat{\tau}[v]$ 和 $\hat{\tau}[w]$ 发生变化的量. 具体而言, 当 $\sigma\in\mathcal{A}^t$ 时, $\delta_\sigma=+1$; 当 $\sigma\in\mathcal{D}^t$ 时, $\delta_\sigma=-1$. 假设在时间 s, 边 $\{u,v\}$ 被添加或删除. 如果边 $\{v,w\}$ 和 $\{u,w\}$ 被哈希函数 $h(\cdot)$ 映射到同一个工作节点, 那么:

$$\gamma_\sigma=\frac{m^2}{|\mathcal{W}|}\delta_\sigma,\tag{19}$$

否则 $\gamma_\sigma=0$. 根据 γ_σ 和 $\hat{\tau}^t=\sum_{\sigma\in\mathcal{A}^t\cup\mathcal{D}^t}\gamma_\sigma$ 的定义, 可以得出 $\hat{\tau}^t$ 的方差为:

$$\begin{aligned}Var\left[\hat{\tau}^t\right]=Var\left[\sum_{\sigma\in\mathcal{A}^t\cup\mathcal{D}^t}\gamma_\sigma\right]=&\sum_{\sigma\in\mathcal{A}^t\cup\mathcal{D}^t}\sum_{\omega\in\mathcal{A}^t\cup\mathcal{D}^t}\mathrm{Cov}\left[\gamma_\sigma,\gamma_\omega\right]=\\&\sum_{\sigma\in\mathcal{A}^t\cup\mathcal{D}^t}Var\left[\gamma_\sigma\right]+\sum_{\substack{\sigma,\omega\in\mathcal{A}^t\cup\mathcal{D}^t\\\sigma\neq\omega}}\mathrm{Cov}\left[\gamma_\sigma,\gamma_\omega\right].\end{aligned}\tag{20}$$

式(20)表明 $\hat{\tau}^t$ 的方差可分解为 2 个关键部分. 一部分是各个变化量的方差之和, 另一部分是不同变化量之间的协方差之和. 根据方差的计算公式 $Var[\gamma_\sigma]=\mathbb{E}[(\gamma_\sigma)^2]-(\mathbb{E}[\gamma_\sigma])^2$, 可以得出 γ_σ 的方差为:

$$Var\left[\gamma_\sigma\right]=\frac{m^2}{|\mathcal{W}|}-1. \tag{21}$$

因此,式(20)中的方差项计算为:

$$\sum_{\sigma\in\mathcal{A}^t\cup\mathcal{D}^t}Var\left[\gamma_\sigma\right]=\sum_{\sigma\in\mathcal{A}^t\cup\mathcal{D}^t}\left(\frac{m^2}{|\mathcal{W}|}-1\right)=\left(\left|\mathcal{A}^t\right|+\left|\mathcal{D}^t\right|\right)\left(\frac{m^2}{|\mathcal{W}|}-1\right). \tag{22}$$

对于式(20)中的协方差项,根据协方差的计算公式,可以得出:

$$\mathrm{Cov}[\gamma_\sigma,\gamma_\omega]=\mathbb{E}[\gamma_\sigma\gamma_\omega]-\mathbb{E}[\gamma_\sigma]\mathbb{E}[\gamma_\omega]=\mathbb{E}[\gamma_\sigma\gamma_\omega]-\delta_\sigma\delta_\omega. \tag{23}$$

如果$\sigma,\omega\in\mathcal{A}^t$或$\sigma,\omega\in\mathcal{D}^t$,那么$\delta_\sigma\delta_\omega=1$;如果$\sigma\in\mathcal{A}^t,\omega\in\mathcal{D}^t$或$\sigma\in\mathcal{D}^t,\omega\in\mathcal{A}^t$,那么$\delta_\sigma\delta_\omega=-1$.因此,$\mathbb{E}\left[\gamma_\sigma\gamma_\omega\right]$满足结论:

$$\mathbb{E}\left[\gamma_\sigma\gamma_\omega\right]=\begin{cases}\dfrac{m}{|\mathcal{W}|}, & \text{if } Type_{(\sigma,\omega)}=1 \text{ or } 2,\\ -\dfrac{m}{|\mathcal{W}|}, & \text{if } Type_{(\sigma,\omega)}=3 \text{ or } 4,\\ \dfrac{m^2}{|\mathcal{W}|}, & \text{if } Type_{(\sigma,\omega)}=5 \text{ or } 6,\\ -\dfrac{m^2}{|\mathcal{W}|}, & \text{if } Type_{(\sigma,\omega)}=7 \text{ or } 8,\\ \delta_\sigma\delta_\omega, & \text{if } Type_{(\sigma,\omega)}=9.\end{cases} \tag{24}$$

根据式(23)(24),$\mathrm{Cov}\left[\gamma_\sigma,\gamma_\omega\right]$满足结论:

$$\mathrm{Cov}[\gamma_\sigma,\gamma_\omega]=\begin{cases}\left(\dfrac{m}{|\mathcal{W}|}-1\right), & \text{if } Type_{(\sigma,\omega)}=1 \text{ or } 2,\\ -\left(\dfrac{m}{|\mathcal{W}|}-1\right), & \text{if } Type_{(\sigma,\omega)}=3 \text{ or } 4,\\ \left(\dfrac{m^2}{|\mathcal{W}|}-1\right), & \text{if } Type_{(\sigma,\omega)}=5 \text{ or } 6,\\ -\left(\dfrac{m^2}{|\mathcal{W}|}-1\right), & \text{if } Type_{(\sigma,\omega)}=7 \text{ or } 8,\\ 0, & \text{if } Type_{(\sigma,\omega)}=9.\end{cases} \tag{25}$$

因此,式(20)中的协方差项计算为:

$$\sum_{\substack{\sigma,\omega\in\mathcal{A}^t\cup\mathcal{D}^t\\ \sigma\neq\omega}}\mathrm{Cov}[\gamma_\sigma,\gamma_\omega]=\left(n_1^t+n_2^t-n_3^t-n_4^t\right)\left(\frac{m}{|\mathcal{W}|}-1\right)+\left(n_5^t+n_6^t-n_7^t-n_8^t\right)\left(\frac{m^2}{|\mathcal{W}|}-1\right). \tag{26}$$

因此,将式(22)和式(26)整合到式(20)中,就完成了对式(17)的证明.将$\hat{\tau}^t$替换为$\hat{\tau}^t[u]$,$\mathcal{A}^t$替换为$\mathcal{A}^t[u]$,以及$\mathcal{D}^t$替换为$\mathcal{D}^t[u]$.重复上述过程即可证明式(18). 证毕.

如式(17)和式(18)所示,EHADS算法的显著优势就是它有效地减少了共享边的三角形对引起的协方差.值得注意的是,当$m=|\mathcal{W}|$时,EHADS算法能够完全消除类型1~4的三角形对的协方差.

3.2 EHADS-g

当$|\mathcal{W}|>m$时,即存在大量机器可用于计算图流中三角形数量的估计值时,我们将EHADS算法进行了改进,得到了EHADS-g算法.如图6所示,EHADS-g算法采用了新的Master-Worker-Aggregator架构,具备多个主节点和分层聚合器,而不再是原先的单一主节点和聚合器结构.这样的改进旨在充分发挥集群的分布式处理能力,以提供更为精确的估计值.

EHADS-g算法的伪代码如算法2所示.

算法2: EHADS-g算法($m<|\mathcal{W}|$).

输入:动态图流$\Pi=\left(e^1,e^2,\cdots,e^t\right)$,哈希函数的映射范围$m$;

输出:全局三角形数量的估计值$\hat{\tau}$,每个顶点u的局部三角形数量的估计值$\hat{\tau}[u]$.

主节点:(每个主节点$j\in\mathcal{M}$):

① for each operation $e^t=\{u,v,\delta\}$ from the graph stream Π do
② send $(e^t,h_j(e^t))$ to all workers within the group;
③ end for

工作节点(每个工作节点$i\in\mathcal{W}$):

④ $\mathcal{S}_i\leftarrow\varnothing$;
⑤ $j=\lceil i/m\rceil$;
⑥ for each element $(e^t,h_j(e^t))$ from the master do
⑦ $Update(e^t)$;
⑧ if $\delta=+$ and $h_j(e^t)=i$ then $Insert(\{u,v\})$;
⑨ else $Delete(\{u,v\})$;
⑩ end if
⑪ end for
⑫ function $Update(e^t)$:
⑬ $sum\leftarrow 0$;
⑭ for each common neighbor $w\in\mathcal{N}_i[u]\cap\mathcal{N}_i[v]$ do
⑮ if $\delta=+$ then
⑯ send $(w,1/(q_j[uvw]))$ to secondary aggregator j;
⑰ $sum\leftarrow sum+1/(q_j[uvw])$;
⑱ else
⑲ send $(w,-1/(q_j[uvw]))$ to secondary aggregator j;
⑳ $sum\leftarrow sum-1/(q_j[uvw])$;
㉑ end if
㉒ end for
㉓ send $(\#,sum)$, (u,sum) and (v,sum) to secondary aggregator j;

次聚合节点(每个次聚合节点$j\in C$):

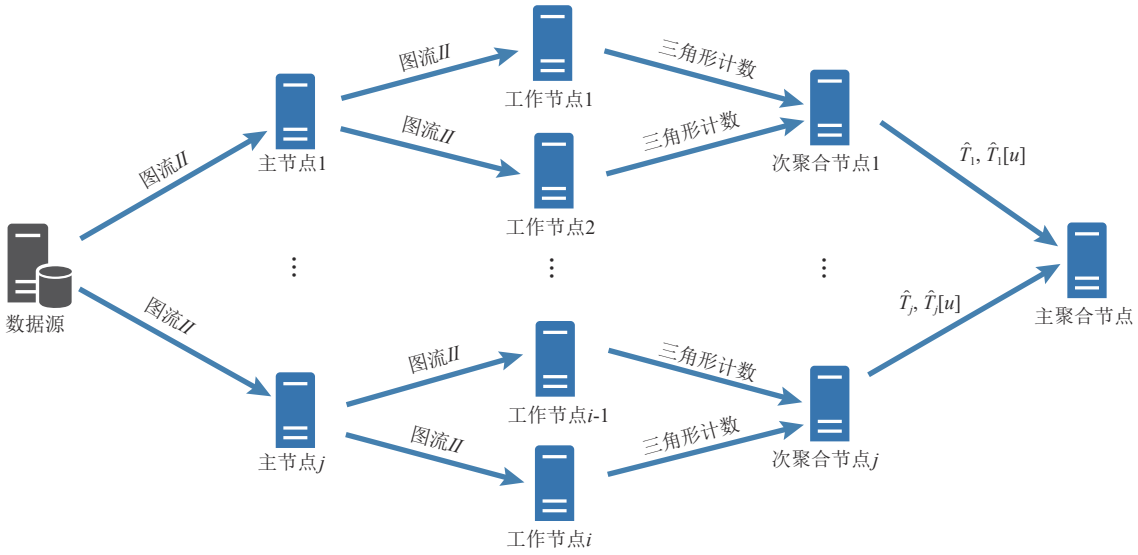


Fig. 6 The framework of EHADS-g algorithm

图 6 EHADS-g 算法框架

㉔ $\hat{T}_j \leftarrow 0, \hat{T}_j[u] \leftarrow 0$;

㉕ $k_1 = \lfloor |\mathcal{W}|/m \rfloor$;

㉖ for each element (u, Δ) from the workers do

㉗ if $u = \#$ then $\hat{T}_j \leftarrow \hat{T}_j + \Delta$;

㉘ else $\hat{T}_j[u] \leftarrow \hat{T}_j[u] + \Delta$;

㉙ end if

㉚ end for

㉛ if $j \leqslant k_1$ then send $\hat{T}_j/k_1$, $\hat{T}_j[u]/k_1$ to primary aggregator;

㉜ else send $\hat{T}_j$, $\hat{T}_j[u]$ to primary aggregator;

㉝ end if

主聚合节点：

㉞ $\hat{\tau} \leftarrow 0, \hat{\tau}[u] \leftarrow 0$;

㉟ $k_1 = \lfloor |\mathcal{W}|/m \rfloor$;

㊱ $k_2 = |\mathcal{W}| \% m$;

㊲ if $k_2 = 0$ then $\hat{\tau} \leftarrow \sum_{j=1}^{k_1} \hat{T}_j, \hat{\tau}[u] \leftarrow \sum_{j=1}^{k_1} \hat{T}_j[u]$;

㊳ else $\hat{\tau} \leftarrow \left(w_1 \sum_{j=1}^{k_1} \hat{T}_j + w_2 \hat{T}_{(k_1+1)} \right) / (w_1 + w_2), \hat{\tau}[u] \leftarrow \left(w_1 \sum_{j=1}^{k_1} \hat{T}_j[u] + w_2 \hat{T}_{(k_1+1)}[u] \right) / (w_1 + w_2)$;

㊴ end if

当 $|\mathcal{W}| > m$ 时，定义 $k_1 = \lfloor |\mathcal{W}|/m \rfloor$ 和 $k_2 = |\mathcal{W}| \% m$，即 $|\mathcal{W}| = k_1 m + k_2$，其中 $k_1 \geqslant 1$ 且 $0 \leqslant k_2 < m$. EHADS-g 算法将 $|\mathcal{W}|$ 个工作节点分成了 $k_1 + 1$ 个组. 前 k_1 组每组包含 m 个工作节点，而最后一组则包含 k_2 个工作节点. 每组工作节点都采用 EHADS 算法. h_j 是第 j 个主节点使用的边哈希函数，其中 $1 \leqslant j \leqslant k_1 + 1$. $h_1, \cdots, h_{k_1+1}$ 彼此之间是相互独立的. 因此，这 $k_1 + 1$ 组工作节点所提供的三角形数量估计值也是相互独立的.

1）当 $k_2 = 0$ 时的算法. 定义每组工作节点提供的估计值为 $\hat{T}_j^t$ 和 $\hat{T}_j^t[u]$，$\hat{T}_j^t$ 和 $\hat{T}_j^t[u]$ 分别满足式（6）和式（7）. 主汇聚节点维护的估计值为 $\hat{\tau}^t$ 和 $\hat{\tau}^t[u]$，并满足：

$$\hat{\tau}^t = \frac{1}{k_1} \sum_{j=1}^{k_1} \hat{T}_j^t, \tag{27}$$

$$\hat{\tau}^t[u] = \frac{1}{k_1} \sum_{j=1}^{k_1} \hat{T}_j^t[u]. \tag{28}$$

根据数学期望的线性性质，可以推导出 $\hat{\tau}^t$ 和 $\hat{\tau}^t[u]$ 是 $\mathcal{G}^t$ 中全局三角形数量和局部三角形数量的无偏估计，即 $\hat{\tau}^t$ 和 $\hat{\tau}^t[u]$ 满足结论：

$$\mathbb{E}\left[\hat{\tau}^t\right] = \mathbb{E}\left[\frac{1}{k_1} \sum_{j=1}^{k_1} \hat{T}_j^t\right] = \frac{1}{k_1} k_1 \left|\mathcal{T}^t\right| = \left|\mathcal{T}^t\right|, \tag{29}$$

$$\mathbb{E}\left[\hat{\tau}^t[u]\right] = \mathbb{E}\left[\frac{1}{k_1} \sum_{j=1}^{k_1} \hat{T}_j^t[u]\right] = \frac{1}{k_1} k_1 \left|\mathcal{T}^t[u]\right| = \left|\mathcal{T}^t[u]\right|. \tag{30}$$

$\hat{T}_j^t$ 和 $\hat{T}_j^t[u]$ 的方差分别满足式（17）和式（18）. 由于 $\hat{T}_j^t$ 之间相互独立，以及 $\hat{T}_j^t[u]$ 之间相互独立，因此 $\hat{\tau}^t$ 和 $\hat{\tau}^t[u]$ 的方差分别为：

$$Var\left[\hat{\tau}^t\right] = Var\left[\frac{1}{k_1} \sum_{j=1}^{k_1} \hat{T}_j^t\right] = \frac{1}{{k_1}^2} \sum_{j=1}^{k_1} Var\left[\hat{T}_j^t\right] = \frac{1}{k_1}(|\mathcal{A}^t| + |\mathcal{D}^t|)(m-1) + \frac{1}{k_1}\left(n_5^t + n_6^t - n_7^t - n_8^t\right)(m-1), \tag{31}$$

$$Var\left[\hat{\tau}^t[u]\right]=Var\left[\frac{1}{k_1}\sum_{j=1}^{k_1}\hat{T}_j^t[u]\right]=\frac{1}{{k_1}^2}\sum_{j=1}^{k_1}Var\left[\hat{T}_j^t[u]\right]=\frac{1}{k_1}\left(\left|\mathcal{A}^t[u]\right|+\left|\mathcal{D}^t[u]\right|\right)(m-1)+\frac{1}{k_1}\left(n_5^t[u]+n_6^t[u]-n_7^t[u]-n_8^t[u]\right)(m-1).\tag{32}$$

2)当$k_2\neq 0$时的算法. 定义前k_1个次聚合节点提供的估计值为$\hat{\tau}_1^t$和$\hat{\tau}_1^t[u]$. $\hat{\tau}_1^t$的数学期望和方差分别满足式(29)和式(31). $\hat{\tau}_1^t[u]$的数学期望和方差分别满足式(30)和式(32). 定义最后一个次聚合节点提供的估计值为$\hat{\tau}_2^t$和$\hat{\tau}_2^t[u]$. $\hat{\tau}_2^t$和$\hat{\tau}_2^t[u]$同样是$\mathcal{G}^t$中全局三角形数量和局部三角形数量的无偏估计. $\hat{\tau}_2^t$的数学期望和方差分别满足式(6)和式(17). $\hat{\tau}_2^t[u]$的数学期望和方差分别满足式(7)和式(18). 主聚合节点维护的估计值$\hat{\tau}^t$和$\hat{\tau}^t[u]$为最终的估计，$\hat{\tau}^t$和$\hat{\tau}^t[u]$满足：

$$\hat{\tau}^t=\frac{w_1\hat{\tau}_1^t+w_2\hat{\tau}_2^t}{w_1+w_2},\tag{33}$$

$$\hat{\tau}^t[u]=\frac{w_1\hat{\tau}_1^t[u]+w_2\hat{\tau}_2^t[u]}{w_1+w_2},\tag{34}$$

其中，w_1和w_2分别表示2个给定的独立且无偏的估计值的权重. 方差较小的估计值应当被赋予更大的权重. 显然，$\hat{\tau}_1^t$的方差小于$\hat{\tau}_2^t$的方差；$\hat{\tau}_1^t[u]$的方差小于$\hat{\tau}_2^t[u]$的方差. 根据数学期望的线性性质，可以得出$\hat{\tau}^t$和$\hat{\tau}^t[u]$的均值为：

$$\mathbb{E}\left[\hat{\tau}^t\right]=\mathbb{E}\left[\frac{w_1\hat{\tau}_1^t+w_2\hat{\tau}_2^t}{w_1+w_2}\right]=\frac{w_1}{w_1+w_2}\mathbb{E}\left[\hat{\tau}_1^t\right]+\frac{w_2}{w_1+w_2}\mathbb{E}\left[\hat{\tau}_2^t\right]=\frac{w_1}{w_1+w_2}\left|\mathcal{T}^t\right|+\frac{w_2}{w_1+w_2}\left|\mathcal{T}^t\right|=\left|\mathcal{T}^t\right|,\tag{35}$$

$$\mathbb{E}\left[\hat{\tau}^t[u]\right]=\mathbb{E}\left[\frac{w_1\hat{\tau}_1^t[u]+w_2\hat{\tau}_2^t[u]}{w_1+w_2}\right]=\frac{w_1}{w_1+w_2}\mathbb{E}\left[\hat{\tau}_1^t[u]\right]+\frac{w_2}{w_1+w_2}\mathbb{E}\left[\hat{\tau}_2^t[u]\right]=\frac{w_1}{w_1+w_2}\left|\mathcal{T}^t[u]\right|+\frac{w_2}{w_1+w_2}\left|\mathcal{T}^t[u]\right|=\left|\mathcal{T}^t[u]\right|.\tag{36}$$

由于$\hat{\tau}_1^t$与$\hat{\tau}_2^t$之间相互独立，以及$\hat{\tau}_1^t[u]$与$\hat{\tau}_2^t[u]$之间相互独立，因此$\hat{\tau}^t$和$\hat{\tau}^t[u]$的方差分别为：

$$Var\left[\hat{\tau}^t\right]=Var\left[\frac{w_1\hat{\tau}_1^t+w_2\hat{\tau}_2^t}{w_1+w_2}\right]=\left(\frac{w_1}{w_1+w_2}\right)^2Var\left[\hat{\tau}_1^t\right]+\left(\frac{w_2}{w_1+w_2}\right)^2Var\left[\hat{\tau}_2^t\right],\tag{37}$$

$$Var\left[\hat{\tau}^t[u]\right]=Var\left[\frac{w_1\hat{\tau}_1^t[u]+w_2\hat{\tau}_2^t[u]}{w_1+w_2}\right]=\left(\frac{w_1}{w_1+w_2}\right)^2Var\left[\hat{\tau}_1^t[u]\right]+\left(\frac{w_2}{w_1+w_2}\right)^2Var\left[\hat{\tau}_2^t[u]\right].\tag{38}$$

3.3 复杂性分析

本节将深入分析EHADS算法的时间和空间复杂性. 假设每个工作节点都采用邻接表在本地内存中存储样本图，这与实验中采用的实现方式一致. 表2中详细列出了EHADS算法中主节点、工作节点和聚合节点的时间复杂度和空间复杂度.

Table 2 Time Complexity and Space Complexity of EHADS Algorithm for Processing the First t Edges in Graph Stream

表2 EHADS算法处理图流中前t条边的时间复杂度和空间复杂度

节点类型	时间复杂度	空间复杂度
主节点	$O(t\lvert\mathcal{W}\rvert)$	$O(1)$
工作节点	$O(t^2/m)$	$O(t/m)$
聚合节点	$O\left(\frac{t^2\lvert\mathcal{W}\rvert}{m}\right)$	$O(\lvert\mathcal{V}^t\rvert)$

3.3.1 时间复杂性分析

定理3. EHADS的时间复杂度. 在EHADS算法处理前t条边操作的情况下，主节点、工作节点和聚合节点的时间复杂度分别为$O(t\lvert\mathcal{W}\rvert)$，$O\left(t^2/m\right)$和$O\left(\frac{t^2\lvert\mathcal{W}\rvert}{m}\right)$.

证明. 在EHADS算法中，主节点需花费$O(1)$时间将边操作e^s发送给工作节点i，其中$s\leqslant t$. 主节点需要将每条边操作发送给所有工作节点. 因此，主节点处理前t条边操作的时间复杂度为$O(t\lvert\mathcal{W}\rvert)$.

在EHADS算法中，工作节点在处理接收到的边操作e^s时，最耗时的步骤是在本地样本图$\mathcal{S}_i$中查找顶点u和v的公共邻居$\mathcal{N}_i[u]\cap\mathcal{N}_i[v]$. 为简化说明，假设$|\mathcal{N}_i[u]|\leqslant|\mathcal{N}_i[v]|$. EHADS算法利用哈希表来实现这一查找过程. 首先，将$\mathcal{N}_i[v]$中的元素插入哈希表，时间复杂度为$O(|\mathcal{N}_i[v]|)$. 接着，对于$\mathcal{N}_i[u]$中的每个元素，在哈希表中查找其是否存在. 若$\mathcal{N}_i[u]$中的元素存在于哈希表中，则该元素为公共邻居. 哈希表中查找一个元素的平均时间复杂度为$O(1)$. 因此，根据给定哈希表查找交集的时间复杂度为$O(|\mathcal{N}_i[u]|)$. 综上所述，在处理当前输入边操作e^s时，工作节点寻找顶点u和v公共邻居$\mathcal{N}_i[u]\cap\mathcal{N}_i[v]$的总体时间复杂度为$O(|\mathcal{N}_i[u]|+|\mathcal{N}_i[v]|)$. 此外，该时间复杂度满足：

$$O(\mathbb{E}[|\mathcal{N}_i[u]|+|\mathcal{N}_i[v]|])=O\left(\mathbb{E}\left[\left|\mathcal{S}_i^s\right|\right]\right)=O\left(s\frac{1}{m}\right).\tag{39}$$

在式(39)中，$(|\mathcal{N}_i[u]|+|\mathcal{N}_i[v]|)$在通常情况下远小于$\left|\mathcal{S}_i^s\right|$. 根据式(39)，可以得出每个工作节点在处理边操作e^s时的平均时间复杂度为$O\left(s\frac{1}{m}\right)$. 因此，每个工

作节点处理前t条边操作的时间为$O\left(\sum_{s=1}^{t}\frac{s}{m}\right)$. 根据

$$O\left(\sum_{s=1}^{t}\frac{s}{m}\right)=O\left(\frac{1}{m}\frac{t(t+1)}{2}\right)=O\left(\frac{t^2}{m}\right), \tag{40}$$

能够得出每个工作节点处理前t条边操作的平均时间复杂度为$O(t^2/m)$.

处理每条边操作时，工作节点需要向聚合节点发送$(|\mathcal{N}_i[u]\cap\mathcal{N}_i[v]|+3)$次计数结果. 在处理边操作$e^s$时，$(|\mathcal{N}_i[u]\cap\mathcal{N}_i[v]|+3)$满足：

$$\mathbb{E}[|\mathcal{N}_i[u]\cap\mathcal{N}_i[v]|+3]<\mathbb{E}\left[\left|\mathcal{S}_i^s\right|+3\right]<\frac{s}{m}+3. \tag{41}$$

由式(41)可得，算法在处理边操作e^s时，所有工作节点向聚合节点发送计数结果的次数不超过$\sum_{i\in\mathcal{W}}\left(\frac{s}{m}+3\right)$. 因此，算法在处理前$t$条边时，所有工作节点向聚合节点发送计数结果的次数不超过$\sum_{i\in\mathcal{W}}\sum_{s=1}^{t}\left(\frac{s}{m}+3\right)$. 聚合节点处理每个接收到的计数结果的时间复杂度为$O(1)$. 根据

$$\sum_{i\in\mathcal{W}}\sum_{s=1}^{t}\left(\frac{s}{m}+3\right)=\sum_{i\in\mathcal{W}}\left(\frac{t(t+1)}{2m}+3t\right)=\frac{t(t+1)|\mathcal{W}|}{2m}+3t|\mathcal{W}|=O\left(\frac{t^2|\mathcal{W}|}{m}\right). \tag{42}$$

可以得出聚合节点处理前t条边操作的平均时间复杂度为$O\left(\frac{t^2|\mathcal{W}|}{m}\right)$. 证毕.

3.3.2 空间复杂性分析

表2中详细列出了EHADS算法中主节点、工作节点和聚合节点的空间复杂度. 在EHADS算法中，主节点只需要$O(1)$的空间，因为它只负责将从数据源接收到的边操作发送给工作节点. 在处理图流中前t条边操作的情况下，每个工作节点平均需要在本地存储$|\mathcal{E}^t|/m$条样本边. 因此，EHADS算法中工作节点的平均空间复杂度为$O(t/m)$. 如果算法引入了惰性聚合策略，那么工作节点需要额外的空间来暂时存储全局三角形数量的1个估计值和至多$|\mathcal{V}^t|$个局部三角形数量的估计值. 这将增加工作节点的平均空间复杂度至$O(t/m+|\mathcal{V}^t|)$. 聚合节点只需要$O(|\mathcal{V}^t|)$的空间，用于保存一个全局三角形数量的估计值和$|\mathcal{V}^t|$个局部三角形数量的估计值.

4 实验验证

在本节中，我们在6个公开的真实世界图数据集上对EHADS算法的性能进行测试，旨在回答5个问题：

1)无偏性(Q1)：EHADS是否能够提供具有更小方差的无偏估计？

2)准确性(Q2)：与最先进的流算法相比，EHADS的估计准确性是否有所提升？

3)工作节点数量的影响(Q3)：工作节点数量对这些算法性能的影响如何？

4)速度(Q4)：相较于之前的流算法，EHADS的运行速度如何？

5)可扩展性(Q5)：EHADS的运行时间是否与输入图流中的边操作数量呈线性扩展关系？

4.1 实验设置

在实验中，排除了图中的自环和每条边的方向. 实验中使用的数据集均可以从斯坦福网络分析平台(SNAP)获取. 我们针对原始图数据集进行了处理，将其转化为流图. 按照边的输入顺序，为每条边创建了相应的边添加操作. 鉴于原始数据集未包含边的删除信息，我们随机选取了图中20%的边，并在相应边添加操作之后的随机位置引入了边的删除操作. 实验中使用的所有数据集的具体信息如表3所示.

Table 3 Real-World Graph Datasets Used in the Experiments

表3 实验中所用真实世界图数据集

数据集	顶点数	边数	三角形数	描述
Email-Enron	36 692	183 831	727 044	电子邮件通信网络
HepPh	12 008	118 521	3 358 499	合著者网络
AstroPh	18 772	198 110	1 351 441	合著者网络
Gowalla	196 591	950 327	2 273 138	在线社交网络
DBLP	317 080	1 049 866	2 224 385	合著者网络
NotreDame	325 729	1 090 108	8 910 005	网页超链接网络

以下是这些数据集的详细描述：

1)Email-Enron. Enron邮件数据集是一个电子邮件通信网络，其包含约50万封邮件通信. 网络中节点代表电子邮件地址，当地址i至少发送过1封邮件给地址j时，网络中存在从i到j的无向边. 此数据集最初由联邦能源监管委员会公开，并由卡内基梅隆大学的William Cohen发布.

2)HepPh. HepPh数据集源自e-print arXiv，是高能物理现象学合著网络. 该网络记录了在提交到高能物理现象学类别的论文中作者之间的科研合作关系.

3) AstroPh. AstroPh是来自e-print arXiv的天体物理学合著网络. 该网络记录了在提交到天体物理学类别的论文中作者之间的科研合作关系.

4)Gowalla. Gowalla 是一个基于地理位置的社交网络平台. Gowalla 数据集记录了用户在 2009 年 2 月至 2010 年 10 月间的签到行为. 用户通过分享签到记录来展示自己的地理位置信息. 该数据集的友谊网络反映了用户之间的社交连接与互动情况.

5)DBLP. DBLP 是计算机科学领域的文献数据库. DBLP 数据集构建了一个合著者网络, 2 位作者至少合作发表过 1 篇论文才能被连接. 该数据集反映了计算机科学领域内作者之间的合作关系.

6)NotreDame. NotreDame 数据集描述了诺特丹大学网站内部网页之间的连接模式和关系. 其中节点代表网页, 边表示这些网页之间的超链接关系. 由 Albert, Jeong 和 Barabasi 在 1999 年收集整理.

本文的实验在一个包含 32 台机器的集群上执行, 每台机器配备了 2.60GHz Intel® Xeon® E5-2650 v2 CPU 和 128 GB 内存. 我们利用 Scala 编程语言和 Spark 框架实现了 EHADS 算法和一系列基线算法. 这些基线算法是目前最先进的单机流算法, 包括 Triest[52], ThinkD[53], WRS[54] 和 RFES[55] 算法. 在 WRS 算法中, 等待室的相对大小α被设定为 10%. 另外, 值得注意的是, 由于 RFES 算法所需的辅助空间大小通常显著超过预先设定的样本容量数倍乃至数十倍, 因此为确保实验的公平性, EHADS 算法仅在 Q4 实验中与 RFES 算法进行比较. 我们对所有基线算法进行了简单的并行化处理, 使它们在多台机器上独立运行. 对于每个基线算法, 在所有工作节点提供的估计值中取平均值, 以得到最终的估计值. 显然, 简单并行化算法给出的估计值的方差比单机流算法给出的估计值的方差更小. 在实验中, EHADS 算法将与这些简单并行化的流算法进行比较.

4.2 评价指标

我们考虑了输入图流结束时全局三角形和局部三角形的真实数量, 分别用τ和$\{(u,\tau[u])\}_{u\in\mathcal{V}^t}$表示. 同时, 通过流算法得出的全局三角形和局部三角形估计数量分别用$\hat{\tau}$和$\{(u,\hat{\tau}[u])\}_{u\in\mathcal{V}^t}$来表示. 为了评估每个流算法估计三角形数量的准确性, 实验中使用了 2 个指标.

1)全局绝对百分比误差(*GAPE*). 该指标用于衡量全局三角形数量的估计值与真实值之间的差异. 该指标表示为:

$$GAPE=\frac{|\hat{\tau}-\tau|}{\tau}, \tag{43}$$

该值越低越好.

2)均方根误差(*RMSE*). 该指标用于衡量局部三角形数量的估计值$\{(u,\hat{\tau}[u])\}_{u\in\mathcal{V}^t}$与真实值$\{(u,\tau[u])\}_{u\in\mathcal{V}^t}$之间的接近程度. 该指标表示为:

$$RMSE=\sqrt{\frac{1}{|\mathcal{V}^t|}\sum_{u\in\mathcal{V}^t}(\tau[u]-\hat{\tau}[u])^2}, \tag{44}$$

该值越低越好.

在此实验中, 所有算法的每个评估指标均进行了 20 次计算, 并报告了它们的平均值.

4.3 无偏性验证(针对 Q1)

本实验主要验证 EHADS 算法是否能够提供方差较小的无偏估计. 本节选用了 EHADS, WRSDel-Parallel 和 ThinkDfast-Parallel 算法, 并将它们的并行度设置为 10, 每个工作节点的样本容量设置为每个数据集边集大小的 10%. 每个算法独立地进行了 100 次实验, 从而得到了 100 个独立的全局三角形数量估计值. 随后, 利用正态分布函数对这 100 个估计值进行了概率密度函数的拟合. 图 7 展示了 EHADS, WRSDel-Parallel 和 ThinkDfast-Parallel 算法提供的估计量的近似概率密度函数. 结果表明, EHADS 算法提供的估计值平均值与真实的全局三角形数量非常接近, 这与定理 1(EHADS 的无偏性)一致. 此外, 图 7 清晰地显示出 EHADS 算法提供的估计数量更容易接近真实数量, 即 EHADS 提供的估计量的方差相较于 WRSDel-Parallel 和 ThinkDfast-Parallel 更小. 因此, EHADS 算法提供的估计值更为准确.

4.4 准确性验证(针对 Q2)

本实验比较了 EHADS 算法与 4 种简单并行化的流算法在精度上的表现. 每个分布式流算法的工作节点数量均固定为 20, 随后改变工作节点的样本容量, 并且每个工作节点的样本容量不超过数据集边集大小的 25%. 为确保实验结果的可靠性, 每个评估指标均进行了 20 次计算, 并取其平均值.

如图 8 和图 9 所示, 随着工作节点样本容量的增加, 所有分布式流算法在估计全局三角形数量和局部三角形数量方面的误差均呈下降趋势. 在全局三角形数量的估计中, EHADS 算法在 6 个数据集上的估计误差始终保持最低. 在局部三角形数量的估计方面, EHADS 算法同样展现出最小的误差. 值得注意的是, 在 Email-Enron 和 Gowalla 数据集中, 相对于 WRSDel-Parallel 和 ThinkDacc-Parallel 算法, EHADS 算法在局部三角形数量的估计准确性的提升并不显著. 在这 2 个数据集中, 当工作节点的样本容量较低时, EHADS 算法的 RMSE 指标会略高于 WRSDel-Parallel 和 ThinkDacc-Parallel 算法.

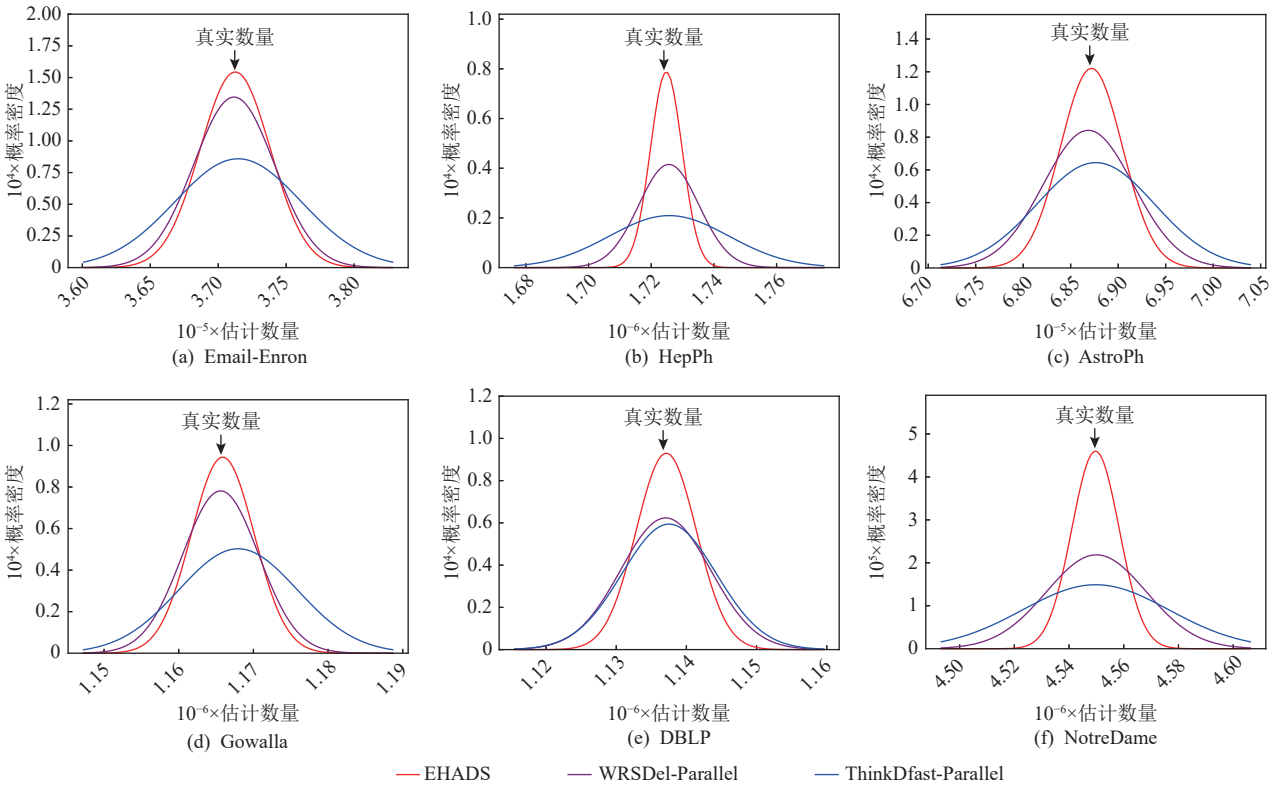


Fig. 7 Approximate probability density function of the estimators provided by different algorithms

图 7 各算法提供的估计量的近似概率密度函数

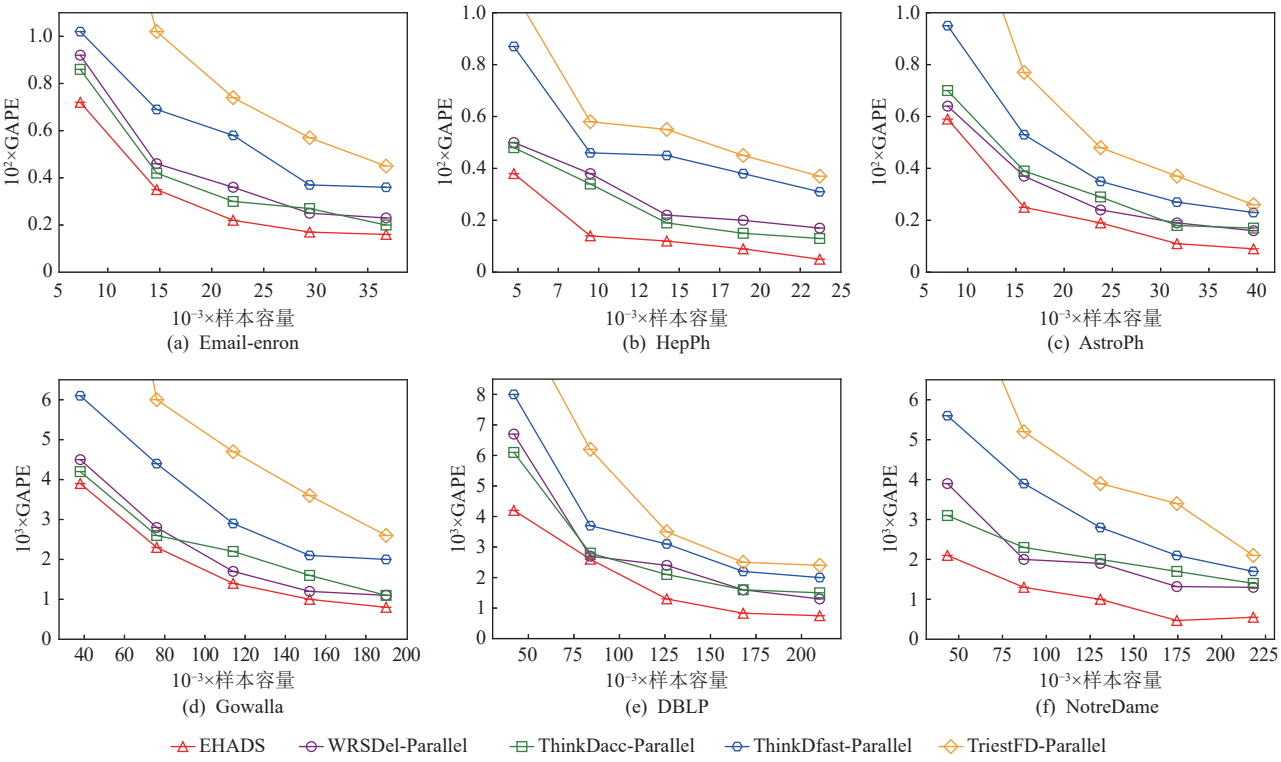


Fig. 8 GAPE of distributed stream algorithms on different datasets

图 8 分布式流算法在不同数据集上的 GAPE

4.5 工作节点数量的影响（针对 Q3）

本实验旨在分析分布式流算法中工作节点数量对性能的影响. 每个分布式流算法的工作节点样本容量被固定为数据集边集大小的 10%, 随后改变工

作节点的数量. 为确保实验结果的可靠性, 每个评估指标都进行了 20 次计算, 并取其平均值.

如图 10 和图 11 所示, 随着工作节点数量的增加, 所有分布式流算法的估计误差都呈现出下降趋势. 在估计全局三角形数量方面, EHADS 算法始终表现出最低的估计误差, 覆盖 6 个不同的数据集. 在估计

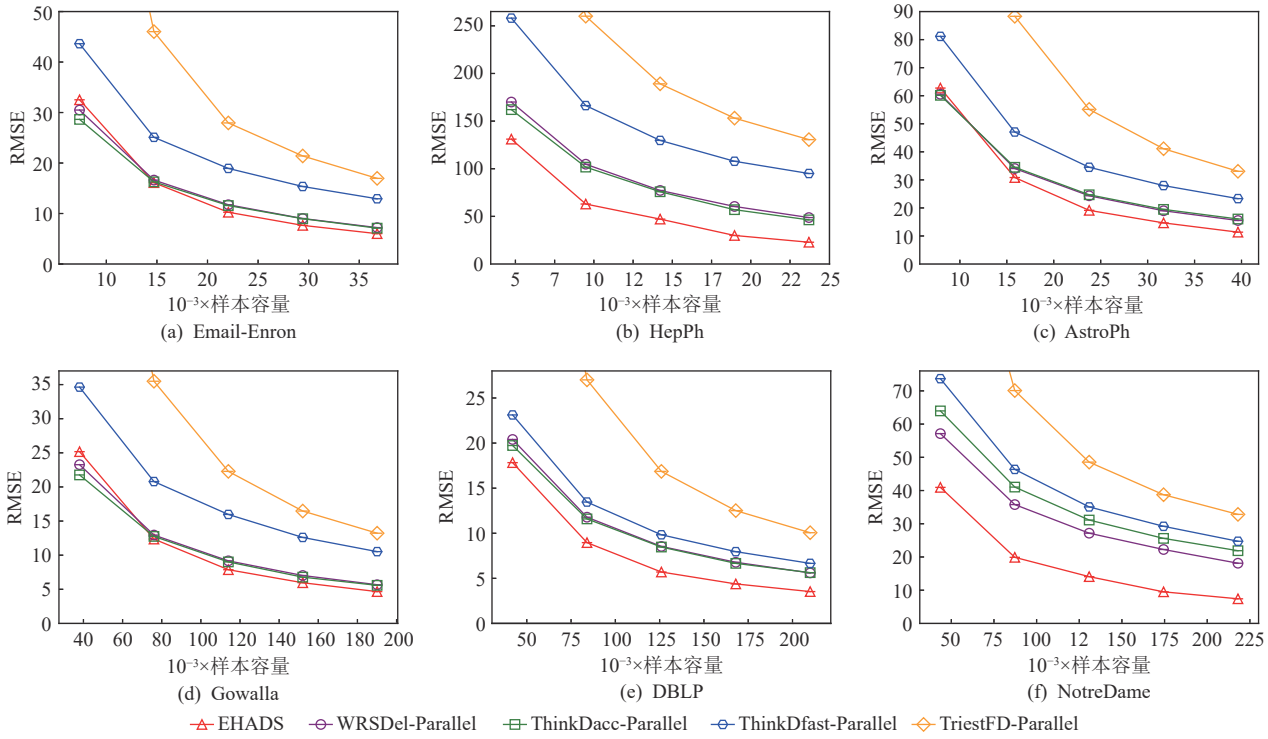


Fig. 9 RMSE of distributed stream algorithms on different datasets

图 9 分布式流算法在不同数据集上的 RMSE

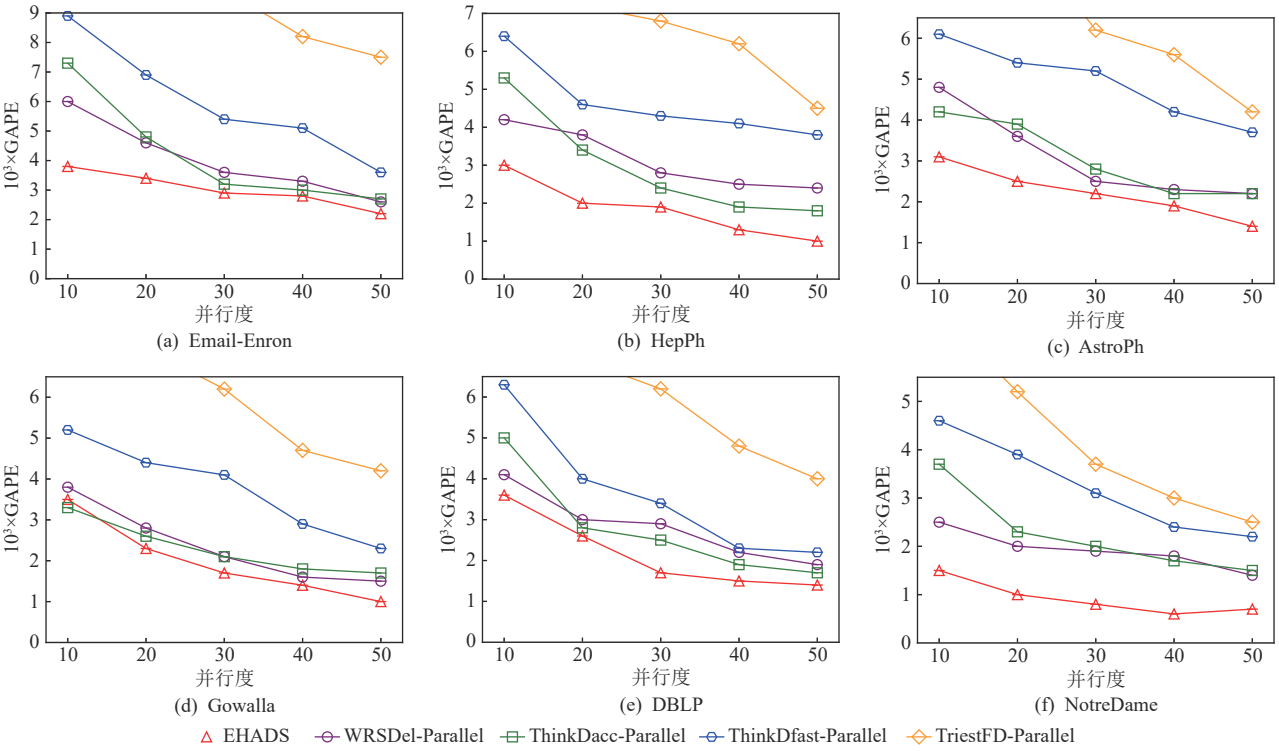


Fig. 10 GAPE of distributed stream algorithms on different datasets

图 10 分布式流算法在不同数据集上的 GAPE

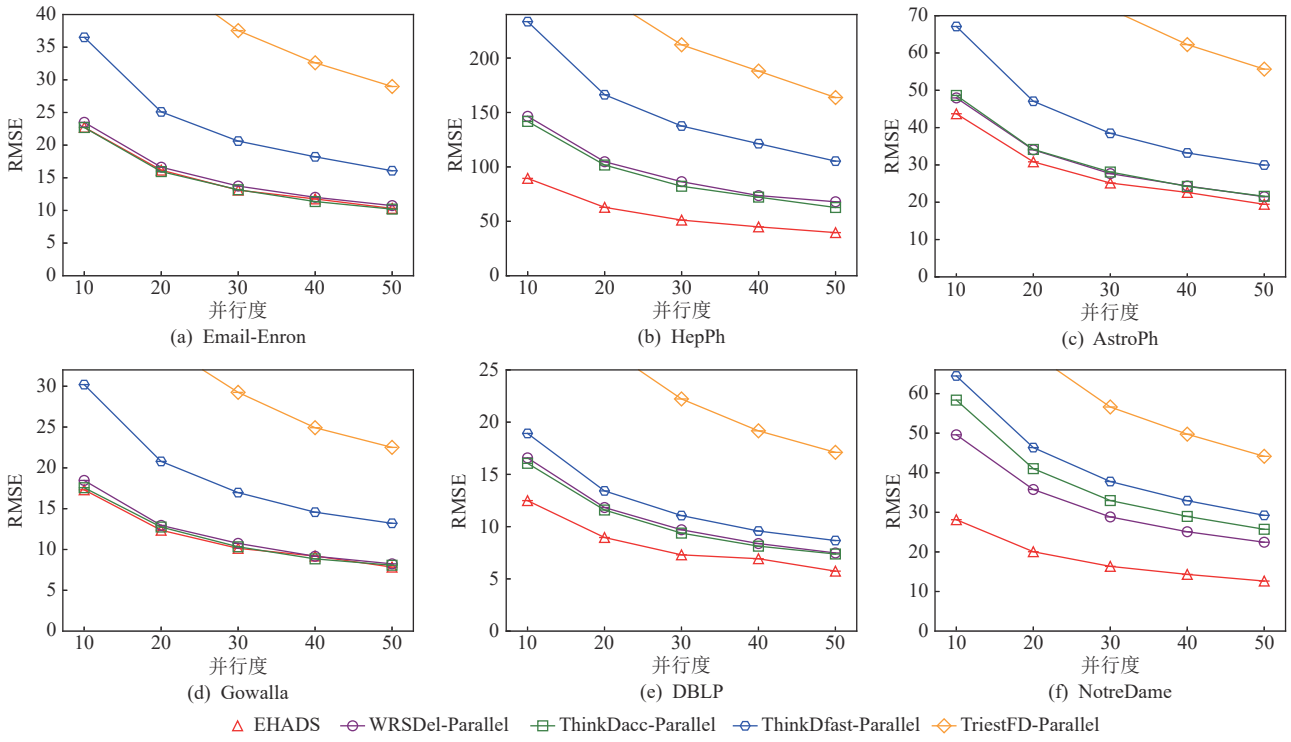


Fig. 11 RMSE of distributed stream algorithms on different datasets

图 11 分布式流算法在不同数据集上的 RMSE

局部三角形数量方面，EHADS 算法同样展现出最小的估计误差. 值得关注的是，在 Email-Enron 和 Gowalla 数据集中，EHADS 算法估计局部三角形数量的准确性与 WRSDel-Parallel 和 ThinkDacc-Parallel 相当.

4.6 运行速度（针对 Q4）

本实验比较了所有分布式流算法的运行时间. 在图 12 所示的实验中，所有分布式流算法的工作节点数量均被设定为 10，每个工作节点的样本容量设置为数据集边集大小的 10%. 图 12 呈现了各算法的运行时间，其中 EHADS 算法在 6 个数据集上均表现出最短的运行时间. EHADS 算法的运行时间与 ThinkDfast-Parallel 相当，明显快于精度最高的 WRSDel-Parallel 算法. 此外，EHADS 算法的运行时间也显著优于 ThinkDacc-Parallel 和 TriestFD-Parallel. 在实验中，所有分布式流算法的工作节点数量均被设定为 10. 表 4 展示了在 GAPE 指标低于 1%（即估计精度达到 99% 时）的情况下，各个分布式流算法的运行时间. 值得注意的是，实验未对算法使用的样本容量进行限制，即所有算法的内存空间使用情况存在差异. 从表 4 可以清晰地看出，在相同的估计精度下，EHADS 算法同样展现出最短的运行时间. 这些比较结果表明，EHADS 算法在保持较高精度的同时，能够更有效地利用计算资

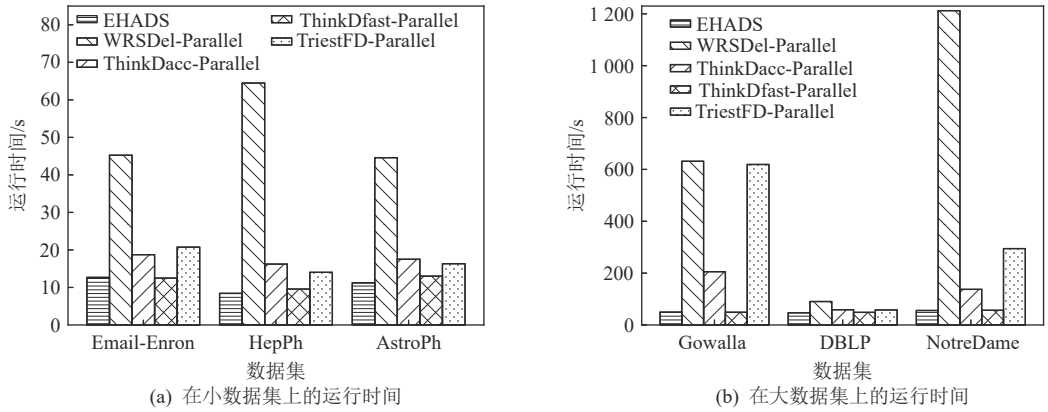


Fig. 12 The runtime of distributed stream algorithms on different datasets

图 12 分布式流算法在不同数据集上的运行时间

Table 4 Runtime of Distributed Stream Algorithms with 99% Estimation Accuracy

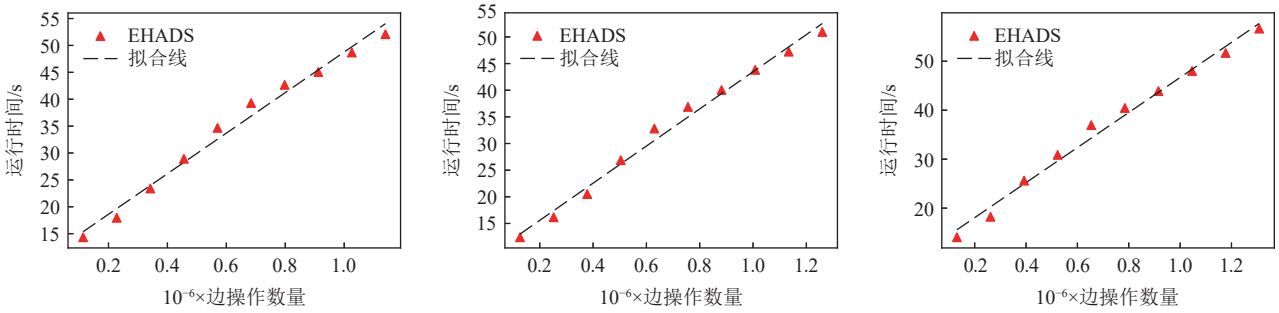
表 4 估计精度为 99% 时分布式流算法的运行时间 s

方法	Email-Enron	HepPh	AstroPh	Gowalla	DBLP	NotreDame
RFESFD-Parallel	22.59	16.93	17.84	683.57	44.55	370.94
TriestFD-Parallel	20.89	21.24	22.65	531.18	55.26	238.02
ThinkDfast-Parallel	14.07	14.32	9.54	45.57	41.19	47.80
ThinkDacc-Parallel	16.24	11.53	13.97	131.76	46.78	96.43
WRSDel-Parallel	25.40	28.95	27.88	143.05	69.19	808.84
EHADS	12.99	9.03	7.09	39.30	33.46	39.14

源, 使其在运行时间上更具优势.

4.7 可扩展性验证 (针对 Q5)

本实验的主要目的是验证 EHADS 算法的可扩展性, 即研究 EHADS 在不同边操作数量下的运行时间变化情况. EHADS 的并行度设置为 10, 每个工作节点的样本容量设置为数据集边集大小的 10%. 对于 Gowalla, DBLP 和 NotreDame 这 3 个数据集, 设置 EHADS 需要处理的边操作数量为 $t = r|\mathcal{E}|$, $r \in \{0.1, 0.2, \cdots, 1.0\}$, r表示边操作数量与数据集边集大小的比例. 如图 13 所示, 能观察到在 Gowalla、DBLP 和 NotreDame 数据集上, EHADS 的运行时间与边操作数量呈现出线性相关的趋势.



(a) Gowalla (b) DBLP (c) NotreDame

Fig. 13 Scalability of EHADS on different datasets

图 13 EHADS 在不同数据集上的可扩展性

5 结论与展望

本文提出了 EHADS 算法, 它是一个用于估计动态流式图中全局和局部三角形数量的分布式流算法. 理论分析表明, EHADS 算法有效地减小了共享边的三角形对引起的协方差. 通过对 6 个真实世界图数据集进行大量实验, 可以观察到在相同的样本容量下, EHADS 算法相对于简单并行化的流算法, 具有更高的估计精度和更快的运行速度. 此外, EHADS 算法还实现了工作负载平衡. 尽管这些结果对于解决动态流式图中的三角形估计问题具有重要意义, 但 EHADS 算法仍然存在一些改进空间, 需要在未来的研究中深入探讨:

1) EHADS-g 算法中参数w_1和w_2的确定. 在 EHADS-g 算法中, 当$k_2 \neq 0$时, 权重w_1和w_2的取值会直接影响线性组合后的估计值的误差. 在当前方法中, w_1和w_2的取值是人为设置的, 并未提供 2 个独立且无偏估计值的最优线性组合. 后续工作可以进一步研究如何更准确地确定w_1和w_2.

2) 样本容量的有效利用. EHADS 算法尚未充分利用给定的样本容量, 其每个工作节点的样本大小随着处理边数量的增加而增长. 未来的研究可以探讨引入随机配对(random pairing)抽样技术的可能性, 以更有效地利用样本容量. 通过在样本容量未满时将每条到达的边添加至样本, 可以提高对三角形的采样概率, 从而降低无偏估计的方差.

3) 算法的实际应用. 尽管本文已验证了算法的有效性, 但其实际应用尚未展开. 未来的工作可以将 EHADS 算法应用于实际问题, 以评估其在真实应用场景中的性能和适用性.

作者贡献声明: 何玉林负责总体构想、撰写论文并对论文进行审读与修订; 吴波负责方法设计和有效验证、参与论文撰写并对论文进行审读与修订; 吴定明负责调查研究, 并审读和修订论文; 黄哲学审读与修订论文; Philippe Fournier-Viger 参与论文的审读与修订.

参 考 文 献

[1] Chung F R K, Lu L. Complex graphs and networks[M]. Rhode Island: American Mathematical Soc, 2006

[2] Kwak H, Lee C, Park H, et al. What is Twitter, a social network or a news media?[C]//Proc of the 19th Int Conf on World Wide Web. New York: ACM, 2010: 591−600

[3] Newman M E J. Coauthorship networks and patterns of scientific collaboration[J]. Proceedings of the National Academy of Sciences, 2004, 101(suppl_1): 5200−5205

[4] Ni Qingjian, Peng Wenqiang, Zhang Zhizheng, et al. Spatial-temporal graph neural network for traffic flow prediction based on information enhanced transmission[J]. Journal of Computer Research and Development, 2022, 59(2): 282−293 (in Chinese)
(倪庆剑, 彭文强, 张志政, 等. 基于信息增强传输的时空图神经网络交通流预测[J]. 计算机研究与发展, 2022, 59(2): 282−293)

[5] Girvan M, Newman M E J. Community structure in social and biological networks[J]. Proceedings of the National Academy of Sciences, 2002, 99(12): 7821−7826

[6] Aggarwal C C. An Introduction to Social Network Data Analytics[M]. Berlin: Springer, 2011

[7] Watts D J, Strogatz S H. Collective dynamics of ‘small-world’ networks[J]. Nature, 1998, 393(6684): 440−442

[8] Newman M E J. The structure and function of complex networks[J]. SIAM Review, 2003, 45(2): 167−256

[9] Newman M E J, Watts D J, Strogatz S H. Random graph models of social networks[J]. Proceedings of the National Academy of Sciences, 2002, 99(suppl_1): 2566−2572

[10] Batagelj V, Zaveršnik M. Short cycle connectivity[J]. Discrete Mathematics, 2007, 307(3): 310−318

[11] Burt R S. Structural holes and good ideas[J]. American Journal of Sociology, 2004, 110(2): 349−399

[12] Foucault Welles B, Van Devender A, Contractor N. Is a Friend a Friend? Investigating the Structure of Friendship Networks in Virtual Worlds[M]//CHI’10 Extended Abstracts on Human Factors in Computing Systems. New York: ACM, 2010: 4027−4032

[13] Zhang F, Gou XY, Zou L. Top-k heavy weight triangles listing on graph stream[J]. World Wide Web, 2023, 26(4): 1827−1851

[14] Yang Z, Wilson C, Wang X, et al. Uncovering social network sybils in the wild[J]. ACM Transactions on Knowledge Discovery from Data, 2014, 8(1): 1−29

[15] Welser H T, Gleave E, Fisher D, et al. Visualizing the signatures of social roles in online discussion groups[J]. Journal of Social Structure, 2007, 8(2): 1−32

[16] Becchetti L, Boldi P, Castillo C, et al. Efficient semi-streaming algorithms for local triangle counting in massive graphs[C]//Proc of the 14th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2008: 16−24

[17] Becchetti L, Boldi P, Castillo C, et al. Efficient algorithms for large-scale local triangle counting[J]. ACM Transactions on Knowledge Discovery from Data, 2010, 4(3): 1−28

[18] Eckmann J P, Moses E. Curvature of co-links uncovers hidden thematic layers in the world wide web[J]. Proceedings of the National Academy of Sciences, 2002, 99(9): 5825−5829

[19] Wang N, Zhang JB, Tan K L, et al. On triangulation-based dense neighborhood graph discovery[J]. Proceedings of the VLDB Endowment, 2010, 4(2): 58−68

[20] Bar-Yossef Z, Kumar R, Sivakumar D. Reductions in streaming algorithms, with an application to counting triangles in graphs[C]// Proc of the 13th annual ACM-SIAM Symposium on Discrete Algorithms. Philadelphia, PA: SIAM, 2002: 623−632

[21] Tsourakakis C E, Drineas P, Michelakis E, et al. Spectral counting of triangles via element-wise sparsification and triangle-based link recommendation[J]. Social Network Analysis and Mining, 2011, 1: 75−81

[22] Epasto A, Lattanzi S, Mirrokni V, et al. Ego-Net community mining applied to friend suggestion[J]. Proceedings of the VLDB Endowment, 2015, 9(4): 324−335

[23] Lim Y, Jung M, Kang U. Memory-efficient and accurate sampling for counting local triangles in graph streams: From simple to multigraphs[J]. ACM Transactions on Knowledge Discovery from Data, 2018, 12(1): 1−28

[24] Latapy M. Main-memory triangle computations for very large (sparse (power-law)) graphs[J]. Theoretical Computer Science, 2008, 407(1): 458−473

[25] Itai A, Rodeh M. Finding a minimum circuit in a graph[C]//Proc of the 9th annual ACM Symposium on Theory of Computing. New York: ACM, 1977: 1−10

[26] Coppersmith D, Winograd S. Matrix multiplication via arithmetic progressions[C]//Proc of the 19th annual ACM Symposium on Theory of Computing. New York: ACM, 1987: 1−6

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

[28] Cohen J. Graph twiddling in a MapReduce world[J]. Computing in Science & Engineering, 2009, 11(4): 29−41

[29] Suri S, Vassilvitskii S. Counting triangles and the curse of the last reducer[C]//Proc of the 20th Int Conf on World Wide Web. New York: ACM, 2011: 607−614

[30] Park H M, Chung C W. An efficient MapReduce algorithm for counting triangles in a very large graph[C]//Proc of the 22nd ACM Int Conf on Information & Knowledge Management. New York: ACM, 2013: 539−548

[31] Arifuzzaman S, Khan M, Marathe M. Patric: A parallel algorithm for counting triangles in massive networks[C]//Proc of the 22nd ACM Int Conf on Information & Knowledge Management. New York: ACM, 2013: 529−538

[32] Park H M, Silvestri F, Kang U, et al. MapReduce triangle enumeration with guarantees[C]//Proc of the 23rd ACM Int Conf on Information and Knowledge Management. New York: ACM, 2014: 1739−1748

[33] Park H M, Myaeng S H, Kang U. Pte: Enumerating trillion triangles on distributed systems[C]//Proc of the 22nd ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2016: 1115−1124

[34] Park H M, Silvestri F, Pagh R, et al. Enumerating trillion subgraphs on

distributed systems[J]. ACM Transactions on Knowledge Discovery from Data, 2018, 12(6): 1−30

[35] Tsourakakis C E, Kang U, Miller G L, et al. Doulion: Counting triangles in massive graphs with a coin[C]//Proc of the 15th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2009: 837−846

[36] Pagh R, Tsourakakis C E. Colorful triangle counting and a MapReduce implementation[J]. Information Processing Letters, 2012, 112(7): 277−281

[37] Ahmed N K, Duffield N, Neville J, et al. Graph sample and hold: A framework for big-graph analytics[C]//Proc of the 20th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2014: 1446−1455

[38] Lim Y, Kang U. MASCOT: Memory-efficient and accurate sampling for counting local triangles in graph streams[C]//Proc of the 21st ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2015: 685−694

[39] Ahmed N K, Duffield N, Willke T, et al. On sampling from massive graph streams[J]. arXiv preprint arXiv: 1703.02625, 2017

[40] Shin K, Hammoud M, Lee E, et al. Tri-Fly: Distributed estimation of global and local triangle counts in graph streams[C]//Proc of Pacific-Asia Conf on Knowledge Discovery and Data Mining. Berlin: Springer, 2018: 651−663

[41] Wang PH, Jia P, Qi YY, et al. REPT: A streaming algorithm of approximating global and local triangle counts in parallel[C]//Proc of the 35th Int Conf on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2019: 758−769

[42] Shin K, Lee E, Oh J, et al. CoCos: Fast and accurate distributed triangle counting in graph streams[J]. ACM Transactions on Knowledge Discovery from Data, 2021, 15(3): 1−30

[43] Yang X, Song C, Yu MD, et al. Distributed triangle approximately counting algorithms in simple graph stream[J]. ACM Transactions on Knowledge Discovery from Data, 2022, 16(4): 1−43

[44] Gou XY, Zou L. Sliding window-based approximate triangle counting with bounded memory usage[J]. The VLDB Journal, 2023, 32(5): 1087−1110

[45] Jha M, Seshadhri C, Pinar A. A space efficient streaming algorithm for triangle counting using the birthday paradox[C]//Proc of the 19th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2013: 589−597

[46] 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

[47] Tangwongsan K, Pavan A, Tirthapura S. Parallel triangle counting in massive streaming graphs[C]//Proc of the 22nd ACM Int Conf on Information & Knowledge Management. New York: ACM, 2013: 781−786

[48] Zhang LL, Jiang H, Wang F, et al. T-sample: A dual reservoir-based sampling method for characterizing large graph streams[C]//Proc of the 35th Int Conf on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2019: 1674−1677

[49] Kutzkov K, Pagh R. Triangle counting in dynamic graph streams[C]// Proc of Scandinavian Workshop on Algorithm Theory. Berlin: Springer, 2014: 306−318

[50] Bulteau L, Froese V, Kutzkov K, et al. Triangle counting in dynamic graph streams[J]. Algorithmica, 2016, 76(1): 259−278

[51] Han G, Sethu H. Edge sample and discard: A new algorithm for counting triangles in large dynamic graphs[C]//Proc of the 2017 IEEE/ACM Int Conf on Advances in Social Networks Analysis and Mining. New York: ACM, 2017: 44−49

[52] Stefani L D, Epasto A, Riondato M, et al. Triest: Counting local and global triangles in fully dynamic streams with fixed memory size[J]. ACM Transactions on Knowledge Discovery from Data, 2017, 11(4): 1−50

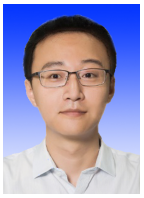
[53] Shin K, Oh S, Kim J, et al. Fast, accurate and provable triangle counting in fully dynamic graph streams[J]. ACM Transactions on Knowledge Discovery from Data, 2020, 14(2): 1−39

[54] Lee D, Shin K, Faloutsos C. Temporal locality-aware sampling for accurate triangle counting in real graph streams[J]. The VLDB Journal, 2020, 29(6): 1501−1525

[55] Yu CY, Liu HM, Wahab F, et al. Global triangle estimation based on first edge sampling in large graph streams[J]. The Journal of Supercomputing, 2023, 79(13): 14079−14116

[56] Vitter J S. Random sampling with a reservoir[J]. ACM Transactions on Mathematical Software, 1985, 11(1): 37−57

[57] Gemulla R, Lehner W, Haas P J. Maintaining bounded-size sample synopses of evolving datasets[J]. The VLDB Journal, 2008, 17: 173−201



He Yulin, born in 1982. PhD, research fellow. Member of CCF. His main research interests include data mining and machine learning, big data processing and analysis, big data approximate computing technology, and multi-sample statistic analysis theory and method.

何玉林, 1982 年生. 博士, 研究员. CCF 会员. 主要研究方向为数据挖掘与机器学习、大数据处理与分析、大数据近似计算技术、多样本统计分析理论与方法.



Wu Bo, born in 2000. Master candidate. His main research interests include social network management and mining, big data system computing technologies, and machine learning algorithms and their applications.

吴　波, 2000 年出生. 硕士研究生. 主要研究方向为社交网络的管理与挖掘、大数据系统计算技术、机器学习算法及其应用.



Wu Dingming, born in 1982. PhD, associate professor. Her main research interest includes the management and mining for spatial text data, knowledge graph data, social network data.

吴定明, 1982 年生. 博士, 副教授. 主要研究方向为空间文本数据、知识图谱数据、社交网络数据上的管理和挖掘.



Huang Zhexue, born in 1959. PhD, professor, PhD supervisor. Member of CCF. His main research interests include data mining, machine learning, big data processing and analysis, and big data system computing technologies.

黄哲学, 1959 年生. 博士, 教授, 博士生导师. CCF 会员. 主要研究方向为数据挖掘、机器学习、大数据处理与分析、大数据系统计算技术.



Philippe Fournier-Viger, born in 1980. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include data mining, artificial intelligence, knowledge expression and inference, and cognitive model construction.

菲律普弗尼尔–维格, 1980 年生. 博士, 教授, 博士生导师. CCF 高级会员. 主要研究方向为数据挖掘、人工智能、知识表示和推理、认知模型构建.