https://crad.ict.ac.cn
62(3): 709−719, 2025

时序图流上的快速子图近似计数算法

王晶晶[1] 王延昊[2] 姜文君[3] 曾一夫[1] 祝团飞[1]
[1](长沙学院计算机科学与工程学院 长沙 410022)
[2](华东师范大学数据科学与工程学院 上海 200062)
[3](湖南大学信息科学与工程学院 长沙 410082)
(wangjingjing2019@hnu.edu.cn;)

Fast Algorithm for Approximate Motif Counting in Temporal Graph Streams

Wang Jingjing[1], Wang Yanhao[2], Jiang Wenjun[3], Zeng Yifu[1], and Zhu Tuanfei[1]
[1](*College of Computer Science and Engineering*, *Changsha University*, *Changsha* 410022)
[2](*School of Data Science and Engineering*, *East China Normal University*, *Shanghai* 200062)
[3](*College of Computer Science and Electronic Engineering*, *Hunan University*, *Changsha* 410082)

Abstract Graphs often have rich temporal information and evolve dynamically over time, which can be modeled as temporal graph streams. A temporal graph stream consists of a set of vertices and a series of timestamped and directed edges, where new vertices and edges arrive continuously over time. Temporal motifs are generalizations of subgraph patterns in static graphs which take into account edge orderings and durations in addition to topologies. Counting the number of occurrences of temporal motifs is a fundamental problem for temporal graph analysis. However, traditional streaming subgraph counting methods cannot support temporal matching, and are only suitable for simple graphs that do not contain temporal information. In addition, existing temporal motifs counting methods suffer from poor performance in temporal graph streams. We thus study approximate temporal motif counting via random sampling in temporal graph streams. We propose a generic streaming edge sampling (SES) algorithm to estimate the number of instances of any temporal motif in a given temporal graph stream. We then provide comprehensive analyses of the theoretical bounds and time complexities of SES. Finally, we perform extensive experimental evaluations for SES on four real world datasets. The results show that SES achieves up to three orders of magnitude speedups over the state-of-the-art sampling methods while having comparable estimation errors for temporal motif counting in the streaming setting.

Key words temporal motif; motif counting; temporal graph streams; random sampling; graph algorithm

摘 要 图数据中包含丰富的时间信息,其拓扑结构随时间动态演变,通常建模为时序图流.时序图流由一组节点和一系列带时间戳的有向边组成,节点、时序边随时间动态增加.其中时序子图是由传统子图模式推广而来,不仅考虑拓扑结构,同时将时序边的顺序和持续时间纳入考量.在时序图流中计算时序子图的出现次数是时序图研究中的一个基础问题.然而,传统流式子图计数方法不支持时序匹配,仅适用于不

收稿日期: 2023-10-10; **修回日期**: 2024-02-06
基金项目: 国家重点研发计划项目(2022ZD0118302); 国家自然科学基金项目(62202169, 62006030, 62302061, 62302062, 62172149, 62172372, 62372067); 湖南省自然科学基金项目(2023JJ40080, 2023JJ40081); 湖南省教育厅科学研究项目(24B0789); 长沙市自然科学基金项目(kq2502339)
This work was supported by the National Key Research and Development Program of China (2022ZD0118302), the National Natural Science Foundation of China (62202169, 62006030, 62302061, 62302062, 62172149, 62172372, 62372067), the Natural Science Foundation of Hunan Province (2023JJ40080, 2023JJ40081), the Scientific Research Foundation of Hunan Provincial Education Department (24B0789), and Changsha Natural Science Foundation (kq2502339).

包含时间信息的简单无向图或有向图;并且,现有时序子图计数算法在不断产生新数据的时序图流场景下效率不高.因此,对时序图流上时序子图近似计数问题进行了研究,提出了基于蓄水池采样的流式边采样(streaming edge sampling, SES)算法,并从期望、方差、时间复杂度3个方面对SES算法进行了理论分析.最后,在4个真实数据集上进行了大量实验.实验结果表明,与基线方法相比,SES虽然返回的计数相对误差略大,但计算效率取得了最高3个数量级的大幅提升.

关键词 时序子图;子图计数;时序图流;随机采样;图算法

中图法分类号 TP311

DOI: 10.7544/issn1000-1239.202330788 **CSTR:** 32373.14.issn1000-1239.202330788

如今,图数据越来越多地结合时间信息以描述节点之间的动态关系,这种图被称为时序图[1].时序图通常由一组节点和一系列带有时间戳的有向边表示,这些边被称为时序边.时序图因在互联网、金融、交通等领域的普遍性引起了广泛研究[2-6].基于关系动态演化的特性,图数据可建模为时序图流(temporal graph streams),其中的节点和时序边随时间动态增加.例如,社交网络[2-4]可由时序图流来表示,其中1个用户被表示为1个节点,在某个时刻2个用户间的联系被表示为1条时序边,且节点和时序边随时间不断地产生.类似地,计算机网络传输数据、交易数据等也可以建模为时序图流.

时序图流的一个基本问题是计算时序图流中特定时序子图(temporal motif)的出现次数.该问题具有广泛的实际应用,例如网络特征标识[5,7]、结构预测[8]和欺诈检测等[9].时序图中子图[10-11]的概念在传统子图上进行了拓展.它不仅需要考虑子图结构,还要考虑时间信息,包括时序边的顺序和子图持续时间.例如,图1中M和M'是不同的时序子图.虽然M和M'具有完全相同的拓扑结构,但它们的时序边的顺序是不同的.因此,尽管传统子图计数已得到深入研究,这些方法很难直接应用于时序子图计数(temporal motif counting)问题[5-11].

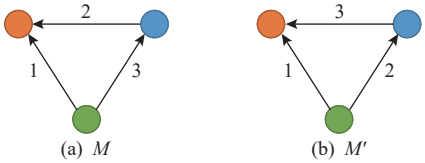


Fig. 1 Examples for temporal motifs

图1 时序子图示例

时序图流中时序子图计数是一项具有挑战性的任务.首先,该问题至少面临传统子图计数相同的困难,其时间复杂度随子图的边数呈指数增长.其次,由于时序子图额外考虑了时间信息,其计算过程更为复杂.例如,传统的时序k-星子图(k-stars)计数存在简单的多项式时间算法;然而,由于时序边顺序的组合性质,时序k-星子图计数问题已被证明是NP难问题[8].然后,时序图是一种多边图,它允许同一对节点之间存在多条不同时间戳的边.因此,在同一组节点中可能存在许多不同的时序子图实例,导致计算复杂度进一步增加.最后,图数据流模型随时间增量地获取节点和边信息,不能事先得到全部数据,且需在任意时刻计算当前已获取数据中时序子图的数目.这些都给计数问题带来更多挑战.

目前已有一些方法可用于时序子图精确计数[5]或枚举[9,12]问题.然而,它们效率低,通常难以扩展到包含数亿条边的大规模时序图[8].另外,文献[8]提出了一种时序子图采样计数方法.它将时序图划分为相等的时间间隔,在采样的时间间隔内,使用精确算法[12]计算子图实例的数目,并根据每个采样间隔里的计数估算总数目.文献[13]提出了基于边采样的时序子图近似计数方法.然而,上述方法面向的是离线时序图,即可提前获取全部数据的时序图,因此,不适用于随时间动态变化的时序图流模型.

本文针对时序图流动态变化的特点,提出了流式时序子图近似计数算法,以处理因规模太大而无法一次性全部装入内存的大规模时序图数据或随时间持续生成的时序图流,分析了算法估计值的期望和方差.本文的主要贡献有3点:

1) 提出了流式边采样(streaming edge sampling, SES)算法估算时序图流中时序子图的实例个数.当1条时序边到达时,SES算法基于蓄水池采样方法实时决定是否采样该条边,如果采样则使用回溯法枚举包含该边的局部子图实例并计数.然后根据所有被采样边的局部计数估算当前时序图流中时序子图的出现次数.

2) 对SES算法返回估计值的期望、方差以及时间复杂度进行深入的理论分析.

3) 在 4 个真实数据集上进行实验，实验结果验证了 SES 算法的有效性. 与最先进的基线方法相比，SES 算法在时序图流处理上更新速度快，实现超过 3 个数量级的加速，并且计算精确度仍具有可比性.

1 相关工作

目前关于时序子图的研究定义了各种类型的时序子图. Viard 等人[14]、Viard 等人[15] 和 Himmel 等人[16] 将最大团的概念扩展到时序图，并提出了有效的最大团枚举算法. Li 等人[6] 提出了持续 k 核的概念以捕捉时序图中社团的持久性. 然而，两者都没有考虑边的顺序. Gurukar 等人[2] 提出了通信子图来表示社交或通信网络中信息传播的频繁模式. Kosyfaki 等人[10] 和 Kovanen 等人[11] 定义了流子图来建模时序图中一个时间窗口内一组节点之间的流传输. Oettershagen 等人[17] 在有标签的时序网络中引入时序子图核用于对传播过程进行分类. 尽管通信子图、流子图和时序子图核都考虑了边的顺序，但定义具有限制性，因为通信子图假设任意 2 条相邻边必须在固定的时间间隔内出现，流子图假设一个子图中的边必须是连续事件，时序子图核针对节点有标签的时序图.

目前已有关于时序子图枚举与计数的研究. Paranjape 等人[5] 首先正式定义了时序子图的概念并提出了精确计数方法，在投影静态图中进行简单子图枚举，再根据时间信息剪枝并精确计数时序子图. Cai 等人[18] 将蝴蝶（butterfly）扩展到了时序网络，并提出时序蝴蝶枚举和计数算法. Kumar 等人[9] 提出了一种高效算法 2SCENT 枚举有向时序图中所有简单的时序环. 2SCENT 被证明对时序环有效，但它不能用于枚举其他类型的时序子图. 潘敏佳等人[19] 提出一种通过添加环路信息来削减搜索空间的新型时序环枚举算法. Mackey 等人[12] 提出了一种有效的时序子图回溯算法. 该算法通过对所有时序子图进行枚举，实现了精确计数. Liu 等人[8] 提出了基于区间的采样的时序子图计数算法. Wang 等人[13] 提出了一种通用的边采样时序子图计数方法以及专门用于 3 个节点、3 条边的边-楔形混合采样算法. 然而，这些算法均需要事先获取离线数据，无法高效应用于图流数据模型.

流式子图计数问题的主要限制是数据动态产生，图的所有信息不能提前保存在内存中，并且数据只能被遍历 1 次. 因此，当新边出现时，流式子图计数算法需要对之前的计算结果进行更新处理，并返回当前图流上的子图计数. 根据各种采样方法使用的内存预算是否固定，现有流式子图计数算法大概可以分为 2 类：基于伯努利采样的方法[20-21] 和基于蓄水池采样的方法[22-25]. 其中伯努利采样要求样本是联合独立的，并且不限制样本大小，因此，数据流中新数据不断产生，可能导致该算法使用内存超过可用内存空间. 而基于蓄水池采样可在内存大小固定的情况下对数据流均匀采样，实现子图计数的无偏估计. 因此基于蓄水池采样的流式子图计数算法得到广泛研究. 本文提出一种流式边采样时序子图近似计数算法，与文献 [5, 8−9, 12−13] 中的算法进行了比较.

Hassan 等人[22] 基于蓄水池采样方法计算流式图的特征向量，捕获图的基本结构进行图分类. Wang 等人[23] 基于蓄水池采样方法提出了一种边加权采样方法 WSD 来估算图流中子图的计数，并且基于强化学习的方法，以数据驱动的方式确定边的权重. Li 等人[24] 利用二分图性质，基于蓄水池采样提出高效采样算法 CAS 用于近似计算二分图流中蝴蝶的数目. Xuan 等人[25] 提出用于计算全局和局部三角形数目的有界采样率算法 BSR-TC，该算法基于蓄水池采样方法，并且可在不断演化的图流上自适应地向上调整内存预算的大小. 但是，这些方法没有考虑时间信息，无法直接应用于时序子图计数.

2 问题定义

本节正式定义了时序图流、时序子图，以及在时序图流上的子图计数问题.

定义 1. 时序图流. 一个时序图流 $\Gamma(t)=(V_{\Gamma(t)}, E_{\Gamma(t)})$ 由包含 n_t 个节点的集合 $V_{\Gamma(t)}$ 和各节点之间的 m_t 条带有时间戳的边序列 $E_{\Gamma(t)}$ 组成，其中时序边的时间戳不大于时间 t. 时序边 $e=(u,v,t)$ 表示在时间 t 生成的从 u 到 v 的一条有向边，其中 $u,v\in V_{\Gamma(t)}$，并且 $t\in\mathbb{R}^+$. $\Gamma(t)$ 中节点、时序边可随时间动态增加.

在不同的时间 t，从 u 到 v 可能有许多时序边. 例如，一个用户可以在 Reddit 上多次评论另一个用户的帖子. 为了便于表示，本文假设每个时序边的时间戳都是唯一的，因此时序边是严格有序的. 需注意的是，若数据中边的时间戳不唯一，可使用一些规则对相同时间戳的边进行排序. 因此，本文方法的应用具有普遍性.

定义 2. 时序子图. 一个时序子图 $M=(V_M,E_M,\sigma)$ 由包含 k 个节点的集合 V_M 和包含 p 条边的集合 E_M 组成的一个连通图，以及 E_M 中边的顺序 σ 构成.

直观地, 一个时序子图 M 可以表示为一个有序的边序列$\langle e'_1=(u'_1,v'_1),e'_2=(u'_2,v'_2),\cdots,e'_p=(u'_p,v'_p)\rangle$. 另外, 在实际社交网络中, 1 h 内形成的子图实例比 1 年内意外形成的实例更有研究价值[2,5]. 所以, 本文只考虑在短时间内形成时序子图的情况. 因此, 给定一个时序图流Γ和一个时序子图 M, 目标是找到一个边序列 $s\subseteq E$, 使得: 1)s 与 M 结构相同; 2)s 与指定的顺序σ相同; 3)s 中的所有边都出现在最大为δ的时间跨度内. 并将这样的边序列 s 称为 M 的一个δ-实例[5,8], 其中 t_p 和 t_1 之间的差称为实例 s 的时长Δs. 下面是正式定义.

定义 3. 子图δ-实例 (motif δ-instance). 时序图流Γ中的一个 p 条边的序列 $s=\langle(w_1,x_1,t_1),(w_2,x_2,t_2),\cdots,(w_p,x_p,t_p)\rangle$, 其中 $(t_1<t_2<\cdots<t_p)$, 是时序子图 $M=\langle(u'_1,v'_1),\cdots,(u'_p,v'_p)\rangle$的$\delta$-实例, 如果: 1)$s$ 和 M 的节点集合之间存在一个双映射函数f, 使得 $f(w_i)=u'_i$并且 $f(x_i)=v'_i$对于所有的$i=1,2,\cdots,p$; 2) 实例 s 的时长Δs最大是δ, 即$t_p-t_1\leqslant\delta$.

例 1. 图 2 中展示时序子图 M_1 及其δ-实例. M_1 的δ-实例不仅与 M_1 完全同构, 而且边的时间戳要满足 $t_1\leqslant t_2\leqslant t_3$, 并且$t_3-t_1\leqslant\delta$.

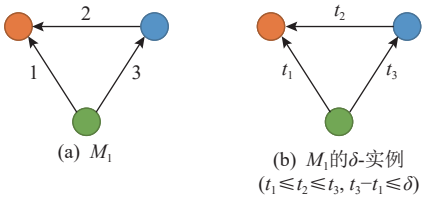


Fig. 2 Temporal motif M_1 and its δ-instance

图 2 时序子图 M_1 及其δ-实例

定义 4. 时序图流中时序子图计数. 给定一个大小未知的时序图流Γ、时序子图M、时间跨度δ, 在任意时刻t, 返回可观测时序图流$\Gamma(t)$中出现的M的δ-实例的数目$C_M(t)$.

算法 1. SES 算法.

输入: 时序图流Γ, 时序子图 M, 时间间隔δ, 蓄水池大小 r;

输出: 时刻 t, $\Gamma(t)$中 M 的δ-实例数估计值$\hat{C}_M(t)$.

① $\hat{E}_\Gamma\leftarrow\varnothing$;

② for e=(u,v,t) do

③ 更新活跃时序边集合$E_\Gamma[t-\delta,t]$;

④ if m_t<r then

⑤ $\hat{E}_\Gamma\leftarrow\hat{E}_\Gamma\cup\{e\}$;

⑥ 将 e 映射到e'_p, 生成一个初始实例 $s^{(1)}$;

⑦ 从 $s^{(1)}$ 开始运行回溯算法, 找到集合 $S(e)$={$s(e)$: $s(e)$ 是 M 的 1 个 δ-实例, 其中 e 映射到e'_p};

⑧ $\eta(e)\leftarrow|S(e)|$;

⑨ return $\hat{C}_M(t)=\sum_{e\in\hat{E}_\Gamma(t)}\eta(e)$;

⑩ else if $random[0,1)<r/m_t$ then

⑪ 从$\hat{E}_\Gamma$选择随机边 g, 并用边 e 替换 g;

⑫ 重复行⑥~⑧;

⑬ return $\hat{C}_M(t)=\frac{m_t}{r}\sum_{e\in\hat{E}_\Gamma(t)}\eta(e)$;

⑭ end if

⑮ end for

3 SES 算法

本节在 3.1 节中详细描述流式边采样算法的过程, 并在 3.2 节中从均值、方差以及时间复杂度 3 方面对算法进行分析.

3.1 算法描述

本文提出一种基于蓄水池采样的 SES 算法, 使其能够在任意时刻t返回时序图流$\Gamma(t)$中时序子图M出现次数$C_M(t)$的估计值$\hat{C}_M(t)$. 蓄水池采样预先设置一个存放采样数据的蓄水池, 并对其进行维护(添加或删除采样数据), 使其在任意时刻都是可观测数据流的一个无偏样本集合. 蓄水池采样主要有 3 方面优点: 首先, 它能够预先设置蓄水池的大小, 对于大小未知的数据流也是如此. 因此, 可在算法运行前为其确定适当的内存大小. 其次, 它只需要在数据流上进行 1 次遍历, 而不需要预先或后期再次处理数据集. 最后, 对于任意时刻的可观测数据流, 它都能均匀采样, 在此基础上能够及时获取最新估计值. 鉴于上述优点, 蓄水池采样已被广泛应用于流式子图计数研究. 然而, 这些工作没有考虑图结构的时间信息, 不能用于处理时序图流数据. 因此, 本文提出 SES 算法.

SES 算法的基本思想为: 通过蓄水池采样从时序图流中均匀采样固定数目的时序边集合, 然后计算采样边的局部时序子图数目, 最后根据每条采样边的局部计数估算可观测时序图流中全局时序子图的数目.

算法 1 给出了 SES 算法的详细实现过程. 给定一个随时间不断增加新边的时序图流Γ、一个时序子图M、一个时间跨度δ和蓄水池大小 r, 初始化蓄水池为空集(行①). 当任意一条时序边 $e=(u,v,t)$到来时, 首先, 更新活跃的时序边集合 $E_\Gamma[t-\delta,t]$(行③), 该集合中只包含时间在$[t-\delta,t]$内的时序边, 是后面进行局部子图计数的搜索空间. 其次, 使用蓄水池采样方法

实时决定是否采样边e(行④~⑭)，并通过添加或删除采样边维护蓄水池$\hat{E}_\Gamma$，详细过程后面将进行介绍. 若e被采样，SES算法通过在$E_\Gamma[t-\delta,t]$上进行精确枚举或近似估算得到包含该边的时序子图M的δ-实例数目(即局部子图数目)$\eta(e)$(该过程后面详细描述). 最后，在时刻t，$\Gamma(t)$中M的δ-实例数目就等于所有采样边的局部计数的和除以采样率$\max\{1,r/m_t\}$，即$\hat{C}_M(t)=\sum_{e\in\hat{E}_\Gamma(t)}\eta(e)$(当$r>m_t$)(行⑨)；$\hat{C}_M(t)=\frac{m_t}{r}\sum_{e\in\hat{E}_\Gamma(t)}\eta(e)$(当$r<m_t$)(行⑬)，其中$\hat{E}_\Gamma(t)$指时刻$t$的采样边集或蓄水池.

SES使用蓄水池采样过程如算法1的行④~⑭所示：当一条时序边$e=(u,v,t)$生成时，如果$m_t\leqslant r$，则采样率为1，将e放入蓄水池；如果$m_t>r$，则以r/m_t的采样率对其进行采样. 具体过程如下：SES生成一个[0,1)之间的随机数，如果随机数小于r/m_t，SES在采样边集$\hat{E}_\Gamma$中随机选择一条边g，并用边e替换g. 实际情况下，$m_t>>r$，因此SES使用蓄水池采样可实现均匀采样，且采样率为r/m_t.

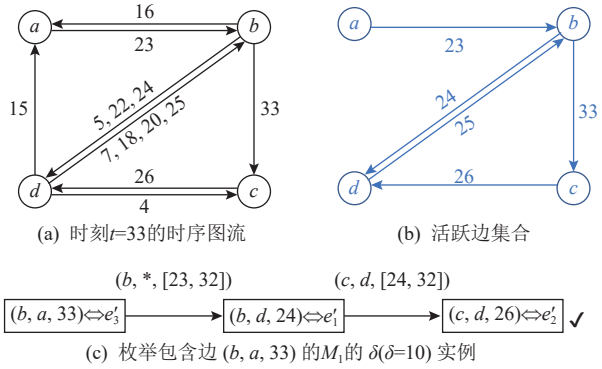
局部子图计数$\eta(e)$的过程为：对于任一采样边$e=(u,v,t)$，SES通过回溯法枚举时序子图M的δ-实例$S(e)$，其中e匹配M中最后一条边e'_p，进而精确计算e的局部子图数目$\eta(e)$. 值得注意的是，该过程是在活跃边集$E_\Gamma[t-\delta,t]$上运行的，由于时序子图的时长约束和图流的特征，此处忽略其他边不影响计数结果. 算法1省略了回溯法的详细过程，因为它基本遵循现有时序子图同构的算法[12-13]，最主要区别在于匹配顺序.

文献[12]的回溯法采用时间优先匹配原则，总是根据M中$\langle e'_1,e'_2,\cdots,e'_p\rangle$的顺序进行匹配. 该方法充分利用时间信息对搜索空间进行剪枝. 文献[13]中ES算法对每一条采样边都运行p次回溯法，分别枚举e匹配$e'_1,e'_2,\cdots,e'_p$时的时序子图，每个时序子图都被枚举p次. 这2种方法都不适用于时序图流. 主要原因是，它们的搜索空间都包含e之后的数据，而时序图流中e之后的数据尚未到来，无法实现匹配，会导致$\eta(e)$计算错误.

为使回溯法适应时序图流，并且提高枚举效率，SES首先匹配M中最后1条边e'_p，其他边的匹配顺序要遵循强制连接性和第1条边优先的原则. 强制连接性指的是，现在待匹配的边必须与一条已匹配的边相邻. 第1条边优先则是指，若有多条待匹配的边满足强制连接性，则M中第1条未匹配的边e'_1将优先进行匹配. 第1个规则可以避免冗余的部分匹配，第2个规则可以限制搜索范围，这2个规则能有效地对搜索空间进行剪枝.

例2. 以图2中M_1为例，确定其匹配顺序，首先匹配M_1中的最后1条边e'_3，然后，根据强制连接性规则可选择e'_1或者e'_2作为第2条匹配边，并根据第2条边优先规则最终选择e'_1. 然后，e'_2作为下一条匹配边. 因此，$\langle e'_3,e'_1,e'_2\rangle$为$M_1$的有效匹配顺序.

例3. 图3展示了使用SES算法在时序图流Γ中枚举M_1的δ-实例($\delta=10$)的过程. 图3(a)是时刻t=33的时序图流Γ，包含了4个节点和13条时序边. 当边$e=(b,a,33)$到来时，根据SES算法，更新活跃边集合$E_\Gamma[23,33]$(图3(b)所示)，集合中包含了在时间范围[23,33]的所有时序边，是后面进行回溯的搜索空间. 图3(c)是使用回溯法为e枚举M_1的10-实例的过程. 每次匹配步骤的状态$(v_{\text{start}},v_{\text{end}},[t_{\text{start}},t_{\text{end}}])$形式给出，其中$v_{\text{start}}$和$v_{\text{start}}$分别是起始节点和结束节点，$[t_{\text{start}},t_{\text{end}}]$是时间范围. 在图3中，“$*$”指可以匹配到任意一个未匹配的节点，并使用“√”表示匹配成功. 按照例2中$\langle e'_3,e'_1,e'_2\rangle$的匹配顺序，SES枚举得到1个实例，因此$\eta(e)=1$.



(a) 时刻t=33的时序图流　(b) 活跃边集合

(c) 枚举包含边$(b,a,33)$的M_1的$\delta(\delta=10)$实例

Fig. 3　Example for SES algorithm

图3　SES算法示例

类似文献[13]，针对3个节点3条边的时序子图，SES进一步优化局部子图计算过程，通过结合楔形采样近似估算采样边的局部子图数目. 首先匹配最后1条边，再采样与该边形成楔形的邻边，计算包含此楔形的时序子图数，在此基础上，估算采样边的局部子图数目.

3.2　理论分析

本节从理论上分析算法1返回的估计值$\hat{C}_M(t)$. 首先在定理1中证明了$\hat{C}_M(t)$是$C_M(t)$的一个无偏估计. 然后在定理2中给出$\hat{C}_M(t)$的方差.

定理1. 由算法1返回的$\hat{C}_M(t)$的期望值$E(\hat{C}_M(t))$是$C_M(t)$.

证明. 给定任意时刻t的时序图流$\Gamma(t)$和蓄水池

大小 r, 首先给可观测的 m_t 条边设置索引编号 $[1,m_t]$, 一般情况下 $m_t > r$, 并使用一个布尔变量 ω_i 来表示第 i 条边 e_i 是否被采样, 即

$$\omega_i = \begin{cases} 1, & e_i \in \hat{E}_T(t), \\ 0, & e_i \notin \hat{E}_T(t), \end{cases}$$

那么, 可以得到

$$\hat{C}_M(t) = \frac{m_t}{r} \sum_{e \in \hat{E}_T(t)} \eta(e) = \frac{m_t}{r} \sum_{i=1}^{m_t} \omega_i \eta(e_i). \tag{1}$$

然后, 基于式(1)和 $E(\omega_i) = \frac{r}{m_t}$, 可得

$$E(\hat{C}_M(t)) = \frac{m_t}{r} \sum_{i=1}^{m_t} E(\omega_i)\eta(e_i) = \sum_{i=1}^{m_t} \eta(e_i) = C_M(t),$$

最后得出结论. 证毕.

定理 2. 算法 1 返回值 $\hat{C}_M(t)$ 的方差值 $Val[\hat{C}_M(t)]$ 最多为 $\frac{m_t - r}{r} C_M^2(t)$.

证明. 根据式(1), 可以得到

$$Val[\hat{C}_M(t)] = Val\left[\sum_{i=1}^{m_t} \frac{m_t \eta(e_i)}{r} \cdot \omega_i\right] = \sum_{i,j=1}^{m_t} \frac{m_t \eta(e_i)}{r} \frac{m_t \eta(e_j)}{r} Cov(\omega_i, \omega_j).$$

因为当 $i \neq j$ 时, 变量 ω_i 和 ω_j 是相互独立的, 所以对于任何 $i \neq j$, 有 $Cov(\omega_i, \omega_j) = 0$. 此外, $Cov(\omega_i, \omega_i) = Val[\omega_i] = \frac{r}{m_t} - \left(\frac{r}{m_t}\right)^2$. 根据以上结果, 可以得到

$$Val[\hat{C}_M(t)] = \sum_{i=1}^{m_t} \frac{m_t^2 \eta^2(e_i)}{r^2} \left(\frac{r}{m_t} - \left(\frac{r}{m_t}\right)^2\right) = \frac{m_t - r}{r} \sum_{i=1}^{m_t} \eta^2(e_i) \leqslant \frac{m_t - r}{r} \left(\sum_{i=1}^{m_t} \eta(e_i)\right)^2 = \frac{m_t - r}{r} C_M^2(t),$$

最后得出结论. 证毕.

根据定理 2 的结果和切比雪夫不等式, 可得 $Pr[|\hat{C}_M(t) - C_M(t)| \geqslant \varepsilon C_M(t)] \leqslant \frac{m_t - r}{r\varepsilon^2}$. 因此, 当 $r = \frac{m_t}{1 + \gamma\varepsilon^2}$ 时, $\hat{C}_M(t)$ 是 $C_M(t)$ 的 (ε, γ) 估计值, 其中 $\varepsilon, \gamma \in (0,1)$.

时间复杂度分析: 首先分析计算一条边 e 的局部计数 $\eta(e)$ 的时间复杂度. 由于 SES 算法中的回溯枚举过程只计算当 e 匹配 e'_p 时的情况, 并且搜索空间最多 $E_T[t-\delta, t]$. 那么, 计算 $\eta(e)$ 的时间复杂度是 $O(d_\delta^{l-1})$, 其中 d_δ 表示与 1 个节点连接的在任意长度时间间隔 δ 内的出边或入边的最大数目. 所以, SES 在 $O\left(\frac{m_t d_\delta^{l-1}}{1 + \gamma\varepsilon^2}\right)$ 时间内提供 $C_M(t)$ 的 (ε, γ) 估计值. SES 比文献 [13] 中 ES 算法的时间复杂度低, 因为每个时序子图 SES 只计数 1 次, 而 ES 计算 p 次.

4 实验评估

本节将在 4 个真实的时序网络数据集上进行大量对比实验, 评估 SES 算法的实验性能. 首先在 4.1 节中介绍实验设置, 然后在 4.2 节分析实验结果.

4.1 实验设置

所有实验都运行在 Ubuntu 18.04.1 LTS 操作系统的服务器上, 该服务器采用 Intel® Xeon® Gold 6140 2.30 GHz 处理器和 250 GB 主内存. 本文使用的所有数据集和代码都是公开. 关于对比算法, 我们下载了原作者发布的代码[5,8-9,12-13], 并按照说明进行编译和使用. 所有算法都是在基于 GCC v7.4 编译器的 C++11 上实现的, 并在单个线程上运行.

实验中使用了 4 种不同的真实世界数据集, 包括 SuperUser(SU), StackOverflow(SO), BitCoin(BC), RedditComments(RC). 所有数据集都是从 SNAP 数据库[26] 等公开平台下载的. 每个数据集都可表示为一个按时间顺序产生的时序边序列. 表 1 中展示了数据集的统计信息, 时间跨度是数据集的总时间跨度.

Table 1 Statistics of Datasets

表 1 数据集统计

数据集	节点数	时序边数	时间跨度/年
SU	192 409	1 108 716	7.60
SO	2 584 164	47 902 865	7.60
BC	48 098 591	113 100 979	7.08
RC	5 688 164	399 523 749	7.44

实验中将所提 SES 算法与下列算法进行比较.

1)EX 是一种时序子图精确计数算法[5], 且仅适用于 3 条边的子图, 不支持有 4 条或更多边的子图(如图 4 中的 Q5).

2)2SCENT 是一种对简单时序环(如图 4 中的 Q4 和 Q5)枚举的算法[9].

3)BT 是用于时序子图同构的一种回溯算法[12]. 它通过枚举所有时序子图来精确计数任意时序子图的数目.

4)IS-BT 是一种基于区间的时序子图采样计数算法[8], 其中 BT [12] 作为子算法用于精确计数 2 个或更多顶点的子图.

5)ES 是基于边采样的时序子图近似计数算法[13].

6)EWS 是结合边采样和楔形采样的近似计数算法[13], 用于计数有 3 个节点和 3 条边的时序子图 (如图 4 中的 Q1~Q4).

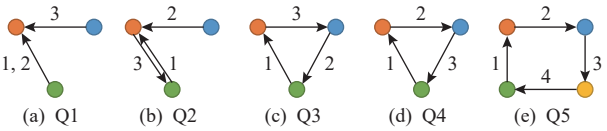


Fig. 4 Query temporal motifs

图 4 查询的时序子图

5 个查询子图如图 4 所示. 一个算法可能不适用于某些子图, 这种情况下将忽略该算法.

算法的效率使用 CPU 时间衡量. 采样算法的精度由相对误差 $\frac{|\hat{x}-x|}{x}$ 衡量, 其中 x 是时序图流中时序子图的精确计数, 而 $\hat{x}$ 是 x 的估计值. 每个实验中, 所有算法将运行 10 次, 并使用平均 CPU 时间和相对误差进行比较.

4.2 实验结果

所有算法的实验结果如表 2 所示. 实验中, 在小数据集 SU 上, 时间间隔 δ 设置为 86 400 s(即 1 天), 在大数据集 SO, BC, RC 上为 3 600 s(即 1 h). 对于 IS-BT, 我们采用该算法的默认设置, 即固定间隔长度为 30δ, 并给出相对误差最多为 5% 的最小区间采样概率的结果. 对于 ES 和 EWS, 我们设置采样率为 0.064, 并且根据原文设置 EWS 的楔形采样率在小数据集 SU 上为 1, 大数据集 SO, BC, RC 上为 0.1. 对于 SES, 设置蓄水池大小为 r=0.064 m, 其中 m 是数据集中时序边的数目.

从表 2 中可以发现, 首先, EX 和 2SCENT 的效率明显低于其他算法. 这是因为它们使用静态图中的子图同构或循环检测算法生成候选子图, 而不考虑边的时间信息, 结果产生了大量违反持续时间约束的冗余候选项, 从而导致性能下降. 其次, 在近似算法中, ES 算法的精确度最高, EWS 算法运行速度比 ES 快, SES 算法在精确度与 ES, EWS 相当的情况下, 效率方面最高达到 3 个数量级的大幅提升. 其中, IS-BT 算法没有明显优势, 大多数情况下其精确度和效率弱于其他近似算法.

通过设置不同的蓄水池大小(样本量) r 值, 测试 r 对算法性能的影响. 由于在综合比较中近似计数算法 IS-BT 的精确度和效率不如 ES 和 EWS 算法, 因此, 后面实验不再对其进行详细比较. 实验中, 对于 SES, 改变蓄水池的大小 r 为 $0.000\,1m$~$0.25m$, 其中 m 是数据集中时序边的数目. 对于 ES 和 EWS, 设置边采样率为 0.000 1~0.25, 从而改变采样边的数目.

实验结果如图 5 和图 6 所示, 分别展示了相对误

Table 2 Running Time and Average Errors of Algorithms

表 2 各算法的运行时间和平均误差

数据集	子图	EX	2SCENT	BT	IS-BT		ES		EWS		SES（本文）	
		时间/ms	时间/ms	时间/ms	误差/%	时间/ms	误差/%	时间/ms	误差/%	时间/ms	误差/%	时间/ms
SU	Q1	3 260	N/A	1 499	3.99	619.7	0.97	561.9	0.97	358.5	1.17	4.536
	Q2			1 650	3.23	670.5	1.48	468.1	1.48	302.6	2.39	4.370
	Q3	4 600	4.6×10^4	1 506	4.85	722.8	3.37	628.1	3.37	2 204	3.91	4.420
	Q4			1 434	3.79	724.5	2.54	681.3	2.54	2 307	4.98	4.538
	Q5	N/A		1 521	4.55	758.9	5.16	312.4	N/A		11.32	4.763
SO	Q1	1.7×10^5	N/A	1.1×10^5	4.82	8 626	0.24	2.7×10^4	0.38	1.0×10^4	0.46	5.334
	Q2			1.1×10^5	4.82	2.7×10^4	0.12	2.4×10^4	0.39	1.1×10^4	0.70	5.595
	Q3	4.7×10^5	2.4×10^5	1.1×10^5	4.30	2.6×10^4	0.37	2.4×10^4	1.35	9 167	2.22	5.589
	Q4			1.1×10^5	4.90	6 775	0.92	2.4×10^4	2.02	9 224	3.18	5.561
	Q5	N/A		9.2×10^4	4.91	9 451	1.75	1.1×10^4	N/A		3.92	5.628
BC	Q1	8.1×10^6	N/A	2.2×10^5	4.75	5.0×10^4	0.33	3.3×10^5	0.34	6.2×10^4	0.76	6.656
	Q2			4.0×10^5	4.90	1.3×10^5	0.37	1.9×10^5	0.38	5.1×10^4	1.47	7.896
	Q3	8.1×10^6	4.7×10^5	4.0×10^5	3.89	9.0×10^4	0.61	2.4×10^5	1.16	1.8×10^4	1.74	6.531
	Q4			4.7×10^5	4.93	9.5×10^4	0.36	2.4×10^5	0.71	2.3×10^4	1.55	6.823
	Q5	N/A		6.0×10^5	4.83	3.2×10^5	1.04	1.3×10^5	N/A		1.29	9.311
RC	Q1	2.8×10^6	N/A	2.0×10^6	4.76	8.4×10^5	1.02	1.2×10^6	0.99	2.0×10^5	3.26	7.546
	Q2			2.1×10^6	4.67	4.3×10^5	0.39	5.0×10^5	0.44	1.7×10^5	0.96	7.428
	Q3	−1	2.2×10^6	2.1×10^6	4.61	7.8×10^5	0.95	3.0×10^5	1.28	9.3×10^4	2.03	7.679
	Q4			1.9×10^6	4.86	6.8×10^5	1.74	3.0×10^5	1.94	9.1×10^4	4.34	7.798
	Q5	N/A		1.6×10^6	4.41	7.1×10^5	3.69	1.5×10^5	N/A		7.23	8.343

注：N/A 表示算法不支持时序子图；−1 表示程序在运行时因内存不足而报错；加下划线数值表示最小误差或最小运行时间.

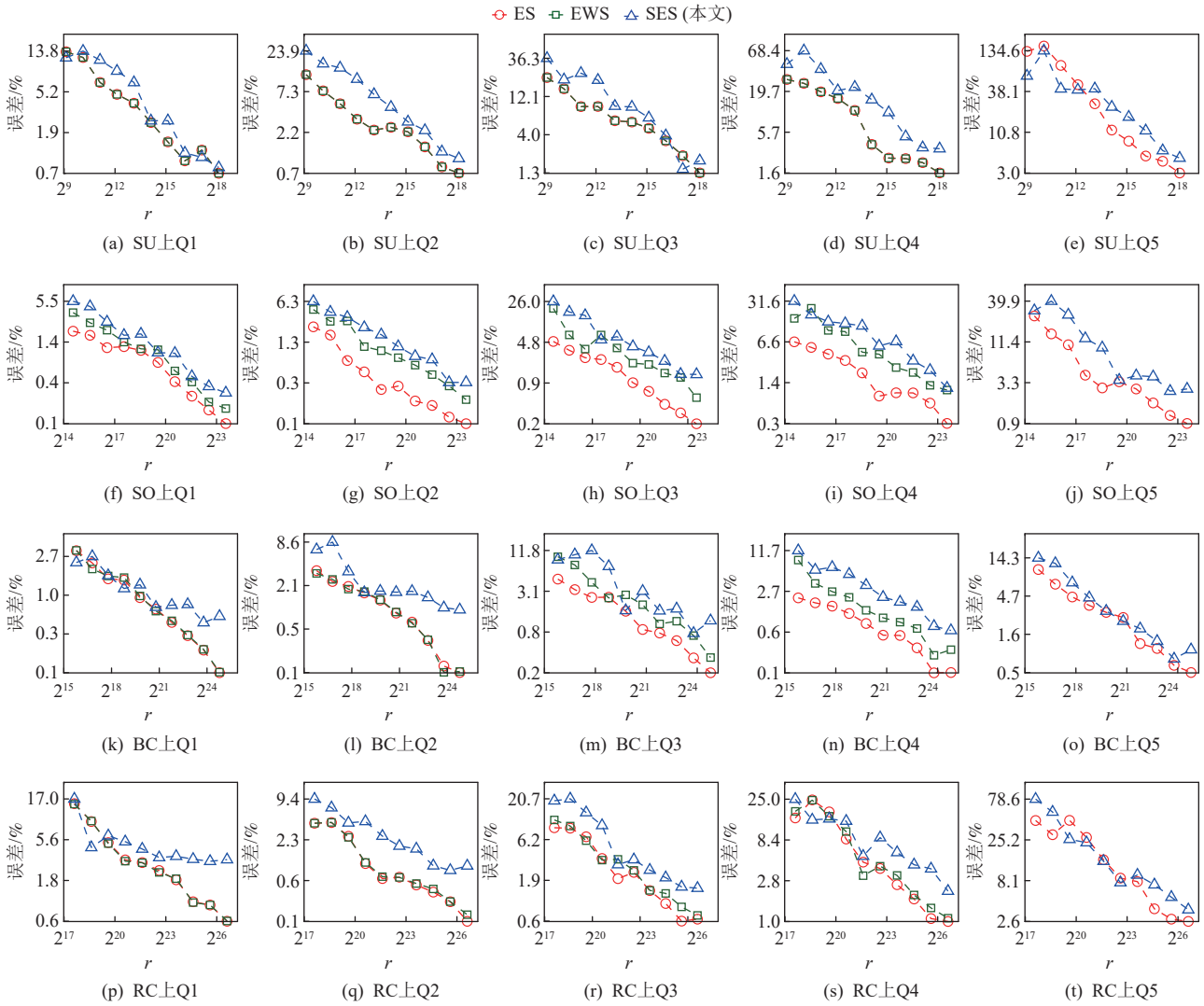


Fig. 5 Relative errors with varying sample size r

图 5 相对误差随蓄水池大小 r 的变化

差和平均更新时间随蓄水池大小 r 的变化, 其中更新时间指处理 1 000 条新增时序边的运行时间.图 5 和图 6 的横轴和纵轴都是采用对数坐标系.从 2 个图可以发现: 1)随着 r 的增加, 所有算法的误差大幅下降, 运行时间增加. 2)算法 ES 和 EWS 的误差比 SES 的稍微小一些. 这是因为在算法 ES 和 EWS 中, 所有的时序子图都被计数 p 次, 其误差相对较小. 而为处理时序图流, SES 将所有的时序子图都只计数 1 次. 3)随着 r 的增加, ES 和 EWS 的更新时间大幅增加, 而 SES 的更新时间基本保持稳定, 计算效率取得了最高 3 个数量级的大幅提升.

随着时间的增加, 图流数据越来越多. 本节在大规模数据集 BC 上测试 SES 算法的性能随时间的变化, 实验设置 r=0.01m. 因图流早期的 r>m_t, 采样率为 1, 估算结果不具有参考性, 所以展示数据后面 24 个月的实验结果. 图 7 展示了时序子图数目随时间的变化, 包括由 BT 算法得到的真实值和 SES 算法的平均估计值以及标准差. 在多数情况下, SES 的平均估计值几乎与真实值重合, 而且标准差比较小.

以在 BC 数据集上查询时序子图 Q3 为例, 通过设置δ为 1~24 h, 测试δ对算法性能的影响, 如图 8 所示.图 8 的横轴采用直角坐标系, 而纵轴采用对数坐标系. 随着δ的增加, 所有算法的误差均有所下降, 每 1 000 条边的平均更新时间则增加. 其中, SES 运行得最快. 当δ增加时, 其更新时间几乎是稳定的; 同时, SES 仍然具有与其他算法相当的相对误差. 证实 SES 在时间跨度δ范围内具有最佳的可扩展性.

5 结束语

本文主要研究大规模时序图流中时序子图近似计数问题. 首先, 提出了基于蓄水池采样的流式时序子

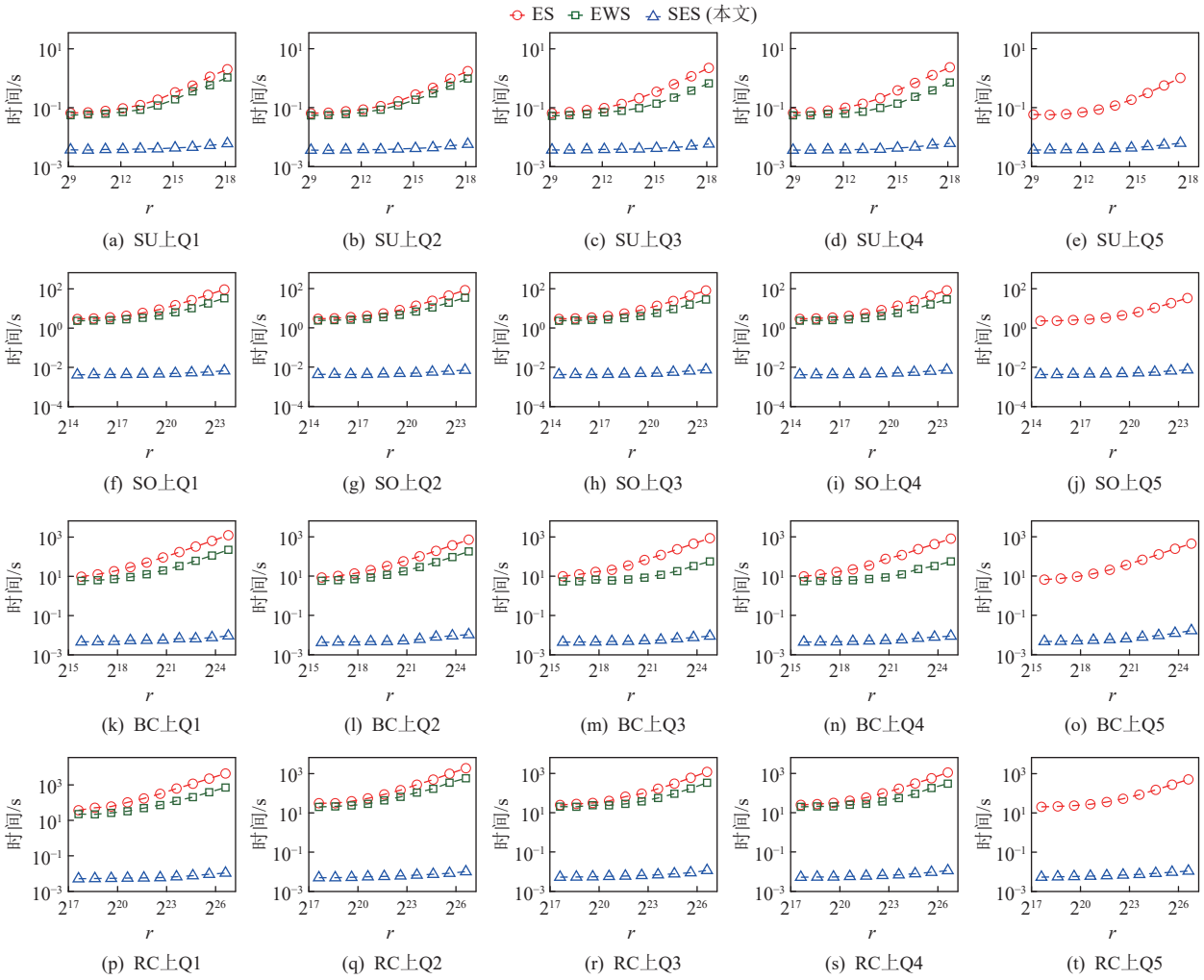
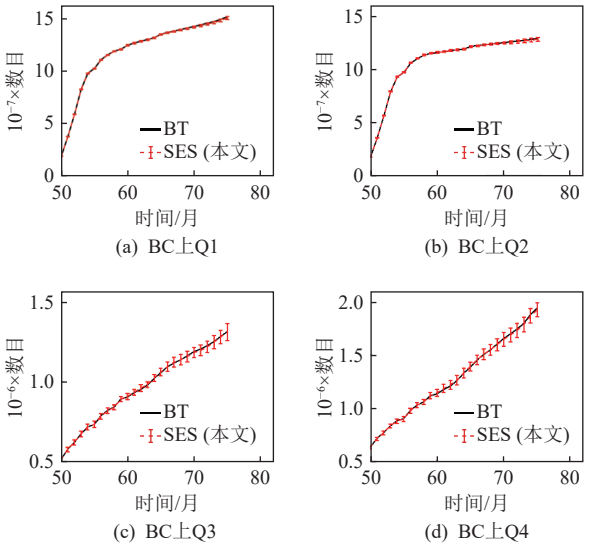


Fig. 6 Average update time per 1 000 edges with varying sample size r

图 6 每 1 000 条边平均更新时间随蓄水池大小 r 的变化



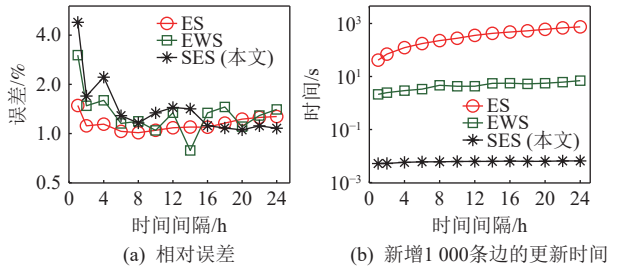


Fig. 8 Query motif Q3 on BC with varying time span δ

图 8 在不同时间间隔δ下查询 BC 上 Q3

Fig. 7 Number of temporal motifs with varying time

图 7 时序子图数目随时间的变化

图近似计数算法；然后对算法估计值的期望、方差和时间复杂度进行了全面的理论分析；最后，在 4 个真实的时序图数据集上进行大量实验，结果表明在处理时序图流数据时，所提算法比其他算法更新速度更快、计算效率更高.

作者贡献声明：王晶晶完成实验设计、实验开发任务并撰写论文；王延昊、姜文君提供实验开发思路，给予指导意见并修改论文；曾一夫、祝团飞完成实验验证任务和提供论文修改意见.

参 考 文 献

[1] Holme P, Saramäki J. Temporal networks[J]. Physics Reports, 2012, 519(3): 97−125

[2] Gurukar S, Ranu S, Ravindran B. Commit: A scalable approach to mining communication motifs from dynamic networks[C]// Proc of the 36th ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2015: 475−489

[3] Zhang Tianming, Zhao Jie, Jin Lu, et al. Vertex betweenness centrality computation method over temporal graphs[J]. Journal of Computer Research and Development, 2023, 60(10): 2383−2393 (in Chinese)
(张天明, 赵杰, 金露, 等. 时态图顶点介数中心度计算方法[J]. 计算机研究与发展, 2023, 60(10): 2383−2393)

[4] Zhang Tianming, Xu Yiheng, Cai Xinwei, et al. A shortest path query method over temporal graphs[J]. Journal of Computer Research and Development, 2022, 59(2): 362−375 (in Chinese)
(张天明, 徐一恒, 蔡鑫伟, 等. 时态图最短路径查询方法[J]. 计算机研究与发展, 2022, 59(2): 362−375)

[5] Paranjape A, Benson A R, Leskovec J. Motifs in temporal networks[C]//Proc of the 10th ACM Int Conf on Web Search and Data Mining (WSDM). New York: ACM, 2017: 601−610

[6] Li Ronghua, Su Jiao, Qin Lu, et al. Persistent community search in temporal networks[C]//Proc of the 34th IEEE Int Conf on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2018: 797−808

[7] Hameed K, Johnston R, Younce B, et al. Motif-based exploratory data analysis for state-backed platform manipulation on Twitter[C]//Proc of the 17th Int AAAI Conf on Web and Social Media. Palo Alto, CA: AAAI, 2023: 315−326

[8] Liu P, Benson A R, Charikar M. Sampling methods for counting temporal motifs[C]//Proc of the 12th ACM Int Conf on Web Search and Data Mining. New York: ACM, 2019: 294−302

[9] Kumar R, Calders T. 2SCENT: An efficient algorithm to enumerate all simple temporal cycles[J]. Proceedings of the VLDB Endowment, 2018, 11(11): 1441−1453

[10] Kosyfaki C, Mamoulis N, Pitoura E, et al. Flow motifs in interaction networks[D]. Ioannina, Greece: University of Ioannina, 2019

[11] Kovanen L, Karsai M, Kaski K, et al. Temporal motifs in time dependent networks[J]. Journal of Statal Mechanics Theory and Experiment, 2011, 11(11): 1−18

[12] Mackey P, Porterfield K, Fitzhenry E, et al. A chronological edge-driven approach to temporal subgraph isomorphism[C]//Proc of the 2018 IEEE Int Conf on Big Data. Piscataway, NJ: IEEE, 2018: 3972−3979

[13] Wang Jingjing, Wang Yanhao, Jiang Wenjun, et al. Efficient sampling algorithms for approximate temporal motif counting[C]//Proc of the 29th ACM Int Conf on Information and Knowledge Management. New York: ACM, 2020: 1505−1514

[14] Viard J, Latapy M, Magnien C. Revealing contact patterns among high-school students using maximal cliques in link streams[C]//Proc of the 2015 IEEE/ACM Int Conf on Advances in Social Networks Analysis and Mining. Piscataway, NJ: IEEE, 2015: 1517−1522

[15] Viard T, Latapy M, Magnien C. Computing maximal cliques in link streams[J]. Theoretical Computer Science, 2016, 609(1): 245−252

[16] Himmel A S, Molter H, Niedermeier R, et al. Enumerating maximal cliques in temporal graphs[C]//Proc of the 2016 IEEE/ACM Int Conf on Advances in Social Networks Analysis and Mining. Piscataway, NJ: IEEE, 2016: 337−344

[17] Oettershagen L, Kriege N M, Jordan C, et al. A temporal graphlet kernel for classifying dissemination in evolving networks[C]// Proc of the 23rd SIAM Int Conf on Data Mining (SDM). Philadelphia, PA: SIAM, 2023: 19−27

[18] Cai Xinwei, Ke Xiangyu, Wang Kai, et al. Efficient temporal butterfly counting and enumeration on temporal bipartite graphs[J]. arXiv preprint, arXiv: 2306.00893, 2023

[19] Pan Minjia, Li Ronghua, Zhao Yuhai, et al. Fast temporal cycle enumeration algorithm on temporal graphs[J]. Journal of Software, 2020, 31(12): 3823−3835 (in Chinese)
(潘敏佳, 李荣华, 赵宇海, 等. 面向时序图数据的快速环枚举算法[J]. 软件学报, 2020, 31(12): 3823−3835)

[20] 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

[21] McGregor A, Vorotnikova S, Vu H T. Better algorithms for counting triangles in data streams[C]//Proc of the 35th ACM SIGMOD-SIGACT-SIGAI Symp on Principles of Database Systems. New York: ACM, 2016: 401−411

[22] Hassan Z R, Ali S, Khan I, et al. Computing graph descriptors on edge streams[J]. ACM Transactions on Knowledge Discovery from Data, 2023, 17(8): 1−25

[23] Wang Kaixin, Long Cheng, Yan Da, et al. Reinforcement learning enhanced weighted sampling for accurate subgraph counting on fully dynamic graph streams[C]//Proc of the 39th IEEE Int Conf on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2023: 1084−1097

[24] Li Rundong, Wang Pinghui, Jia Peng, et al. Approximately counting butterflies in large bipartite graph streams[J]. IEEE Transactions on Knowledge and Data Engineering, 2021, 34(12): 5621−5635

[25] Xuan Wei, Cao Huawei, Yan Mingyu, et al. BSR-TC: Adaptively sampling for accurate triangle counting over evolving graph streams[J]. International Journal of Software Engineering and Knowledge Engineering, 2021, 31(11/12): 1561−1581

[26] Leskovec J, Sosic R. SNAP: A general-purpose network analysis and graph-mining library[J]. ACM Transactions on Intelligent Systems, 2016, 8(1): 1−20



Wang Jingjing, born in 1991. PhD, lecturer. Member of CCF. Her main research interests include graph data mining and social network analysis.

王晶晶, 1991 年生. 博士, 讲师. CCF 会员. 主要研究方向为图数据挖掘、社交网络分析.



Wang Yanhao, born in 1990. PhD, associate professor, PhD supervisor. Member of CCF. His main research interests include algorithm design and analysis, graph data mining, and responsible data management. (yhwang@dase.ecnu.edu.cn)

王延昊, 1990 年生. 博士, 副教授, 博士生导师. CCF 会员. 主要研究方向为算法设计与分析、图数据挖掘、可信数据管理.



Jiang Wenjun, born in 1982. PhD, professor, PhD supervisor. Senior member of CCF. Her main research interests include social network analysis and recommendation system, smart education, and learning optimization.

姜文君, 1982 年生. 博士, 教授, 博士生导师. CCF 高级会员. 主要研究方向为社交网络分析和推荐系统、智慧教育、学习优化.



Zeng Yifu, born in 1990. PhD, master supervisor. Member of CCF. His main research interests include data management, network of data center.

曾一夫, 1990 年生. 博士, 硕士生导师. CCF 会员. 主要研究方向为数据管理、数据中心网络.



Zhu Tuanfei, born in 1987. PhD, associate professor, master supervisor. Member of CCF. His main research interests include ensemble learning and imbalanced data learning.

祝团飞, 1987 年生. 博士, 副教授, 硕士生导师. CCF 会员. 主要研究方向为集成学习、不平衡数据学习.