Memory-efficient and Accurate Sampling for Counting Local Triangles in Graph Streams: From Simple to Multigraphs

YONGSUB LIM, MINSOO JUNG, and U KANG, Seoul National University

How can we estimate local triangle counts accurately in a graph stream without storing the whole graph? How to handle duplicated edges in local triangle counting for graph stream? Local triangle counting, which computes the number of triangles attached to each node in a graph, is a very important problem with wide applications in social network analysis, anomaly detection, web mining, etc.

In this paper, we propose algorithms for local triangle counting in a graph stream based on edge sampling: MASCOT for a simple graph, and MULTIBMASCOT and MULTIWMASCOT for a multigraph. To develop MASCOT, we first present two naïve local triangle counting algorithms in a graph stream, called MASCOT-C and MASCOT-A. MASCOT-C is based on constant edge sampling, and MASCOT-A improves its accuracy by utilizing more memory spaces. MASCOT achieves both accuracy and memory-efficiency of the two algorithms by unconditional triangle counting for a new edge, regardless of whether it is sampled or not. Extending the idea to a multigraph, we develop two algorithms MULTIBMASCOT and MULTIWMASCOT. MULTIBMASCOT enables local triangle counting on the corresponding simple graph of a streamed multigraph without explicit graph conversion; MULTIWMASCOT considers repeated occurrences of an edge as its weight and counts each triangle as the product of its three edge weights. In contrast to the existing algorithm which requires prior knowledge on the target graph and appropriately set parameters, our proposed algorithms require only one parameter of edge sampling probability.

Through extensive experiments, we show that for the same number of edges sampled, MASCOT provides the best accuracy compared to the existing algorithm as well as MASCOT-C and MASCOT-A. We also demonstrate that MULTIBMASCOT on a multigraph is comparable to MASCOT-C on the counterpart simple graph, and MULTIWMASCOT becomes more accurate for higher degree nodes. Thanks to MASCOT, we also discover interesting anomalous patterns in real graphs, including *core-peripheries* in the web, a *bimodal* call pattern in a phone call history, and *intensive collaboration* in DBLP.

CCS Concepts: •**Mathematics of computing** → **Graph algorithms; Approximation algorithms;** •**Information systems** → **Data stream mining;**

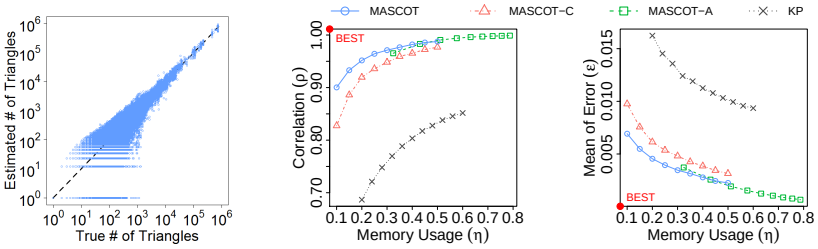
Additional Key Words and Phrases: Local triangle counting; graph stream mining; edge sampling; anomaly detection

1. INTRODUCTION

How can we *count local triangles* in a graph stream with a limited memory space? How accurate are edge sampling strategies for *local triangle counting*? How can we handle repeated occurrences of an edge in the stream? Triangle counting is one of the most widely studied graph mining problems. The number of triangles in a graph becomes an important index indicating cohesiveness of the graph. In many cases, one wants to count triangles adjacent to every node, which helps understand whether the node belongs to a tightly connected group or has diverse neighbors. This problem, called *local triangle counting*, has various applications. For instance, in social networks, triangle counting is used to detect fake accounts [Yang et al. 2014]; the number of triangles of

Corresponding Author: U Kang.
Author's addresses: Y. Lim, M. Jung, and U Kang, Department of Computer Science and Engineering, Seoul National University; emails: yongsub@snu.ac.kr, minsoojung@snu.ac.kr and ukang@snu.ac.kr.

DOI: 0000001.0000001



(a) True vs. Estimation (b) Corr. Coef. (c) Mean of Error

Fig. 1. Summary of our results on the BerkStan graph described in Table III. (a) Scatter plot between the true local triangle counts and its estimations by MASCOT for all nodes. MASCOT is accurate especially for nodes with many triangles. (b), (c): Comparison of our proposed algorithm MASCOT and competitors including KP [Kutzkov and Pagh 2013] for local triangle estimation in a graph stream. Note that for a fixed memory space, MASCOT provides the best accuracy in terms of the correlation coefficient and the error.

a user is examined to identify a social role of the user in the network [Welser et al. 2007], and is shown to be a good feature in assessing the content quality provided by the user [Becchetti et al. 2010]. In web mining, it is also used to find spam pages [Becchetti et al. 2010] and to uncover hidden thematic layers [Eckmann and Moses 2002]. Other applications include network community detection [Berry et al. 2011] and motif detection in bioinformatics [Milo et al. 2002].

Despite enormous bodies of researches on triangle counting, it is still challenging to handle a massive graph due to the complexity of the problem—superlinear time on the graph size is inevitable. Moreover, a number of real graphs appearing especially in recent days are given in a stream fashion whose size is unknown, or even infinite: e.g. packet transmission in the Internet, phone call history, financial transactions, etc. Often, such real graph streams should be analyzed in real time. Thus, designing a streaming algorithm is required for efficient online analysis of a huge graph. A number of algorithms to count triangles in a graph stream have been proposed [Bar-Yossef et al. 2002; Becchetti et al. 2010; Buriol et al. 2006; Jha et al. 2013; Jowhari and Ghodsi 2005; Kane et al. 2012; Pagh and Tsourakakis 2012]. Most of them, however, focus on global triangle counting. Although the first one-pass streaming algorithm for local triangle counting was developed with rigorous theoretical analysis [Kutzkov and Pagh 2013], it is unsuitable for an evolving graph unless an efficient update scheme is explicitly designed. To be used in practice, it also requires knowing the number of nodes in the stream to set hash functions, and a user-defined threshold to count local triangles for nodes having degrees above the threshold. What makes the problem much harder is that real world graph streams involve duplicated edges, which has been rarely focused on in related fields, however.

In this paper, we propose three algorithms for local triangle counting in a graph stream based on edge sampling. The first algorithm is MASCOT for a simple graph; the other two algorithms are MULTIBMASCOT and MULTIWMASCOT for a multigraph which contains duplicated edges. MASCOT provides unbiased estimation of the number of local triangles for every node. We first develop two naïve algorithms, MASCOT-C and MASCOT-A, with simple edge sampling. MASCOT-C is based on constant edge sampling, and MASCOT-A performs MASCOT-C with unconditionally sampling an edge making a triangle. MASCOT-A has lower variance but uses more memory spaces than MASCOT-C. Our proposed algorithm MASCOT provides the advantages of both accuracy and memory-efficiency by the strategy of "unconditional counting before sampling". Table I compares MASCOT and other algorithms including KP [Kutzkov and Pagh 2013] by various aspects. MULTIBMASCOT enables unweighted local triangle

Table I. Comparison of MASCOT and other algorithms. Our main proposed method MASCOT shows the best performance for all metrics. For Corr. Coef. and Error, we compare at the same memory usage, and for the memory usage, we do at the same Error.

	Proposed	Basic		Kutzkov and Pagh [2013]
	MASCOT	MASCOT-A	MASCOT-C	KP
Corr. Coef.	**High**	**High**	Medium	Low
Error	**Small**	Medium	Medium	Large
Memory Usage	**Low**	Medium	Medium	Large
Estimated Node Range	**All**	**All**	**All**	Top-k
Incremental Update	**Yes**	**Yes**	**Yes**	No

counting from a multigraph stream which we call *binary counting* in this paper. This is the same as local triangle counting in real time on the underlying simple graph of the streamed multigraph. MULTIWMASCOT provides weighted local triangle counting from a multigraph stream where duplicated occurrences of an edge are aggregated to its weight. Precisely, the weight of a triangle is determined by the product of the corresponding three edge weights.

Conducting extensive experiments on real world graphs, we show that MASCOT estimates local triangle counts better than MASCOT-C and MASCOT-A in terms of Pearson correlation coefficient and mean of absolute relative error. We also demonstrate that MASCOT outperforms the existing algorithm [Kutzkov and Pagh 2013] in both metrics which to the best of our knowledge is the only one-pass local triangle counting algorithm for a graph stream. Figure 1 illustrates our results for BerkStan graph. In addition, we evaluate MULTIBMASCOT and MULTIWMASCOT. MULTIBMASCOT shows a comparable performance to MASCOT-C which can be used only after converting a multigraph stream to the corresponding simple graph. We examine the accuracy of MULTIWMASCOT with respect to the node degree. The result demonstrates that the accuracy becomes better for higher degree nodes. Applying the proposed algorithms to real graphs, we show that local triangle counts are effective in discovering anomalous patterns in a graph.

Our contributions are summarized as follows.

— **Algorithm.** We propose MASCOT, a memory-efficient and accurate one-pass local triangle counting algorithm for a graph stream. This improves two naïve edge sampling based algorithms MASCOT-C and MASCOT-A to achieve both small variance and memory-efficiency. We also propose MULTIBMASCOT and MULTIWMASCOT to support a multigraph stream. MULTIBMASCOT enables the binary counting (unweighted counting) for a multigraph stream, and MULTIWMASCOT provides weighting triangle counting where each triangle is weighted by the product of its corresponding three edge weights. All of our proposed algorithms require only one simple parameter, the edge sampling probability, without any prior knowledge on the input graph.
— **Performance.** We show that MASCOT is more accurate not only than MASCOT-C and MASCOT-A but also than the previous algorithm [Kutzkov and Pagh 2013] in terms of Pearson correlation coefficient and the mean of absolute relative error. We also show that MULTIBMASCOT on a multigraph stream is comparable to MASCOT-C on the underlying simple graph in accuracy, and MULTIWMASCOT becomes more accurate for nodes having larger degree nodes.

Table II. Table of symbols.

Symbol	Description
G, V, E	Undirected graph, set of its nodes, set of its edges
n, m	Numbers of nodes and edges
$\mathcal{N}_u$	Set of neighbors of a node u
$\mathcal{N}_{uv}$	$\mathcal{N}_u \cap \mathcal{N}_v$
$\mathcal{T}$	Set of triangles in a graph
$\mathcal{T}_u$	Set of triangles of the node u
$\triangle_u$	Number of triangles of the node u, equals to $\lvert\mathcal{T}_u\rvert$
$\tau_u, \tau_u^c, \tau_u^a$	Estimation for $\triangle_u$ in MASCOT, MASCOT-C and MASCOT-A
p	Default probability of sampling each edge
η	Ratio of sampled edges over the total edges
q_{uv}, q_e	True probability that an edge $e = (u, v)$ is sampled
h_e	Weight of an edge e
θ_e	Estimation for h_e

— **Discovery.** We discover several anomalous patterns in real graphs by applying MASCOT, including core-periphery structures in the web, a bimodal calling pattern in a phone call history, and intensive collaboration in DBLP.

The codes and data used in this paper are available at http://datalab.snu.ac.kr/mascot. The rest of this paper is organized as follows. In Section 2, we discuss related works of our paper. We describe our proposed algorithm MASCOT and two naïve ones for a simple graph stream in Section 3, and describe the proposed algorithms MULTIB-MASCOT and MULTIWMASCOT for a multigraph stream in Section 4. After showing the performance of our algorithms in accuracy, and comparing it with the previous algorithm in Section 5, we conclude in Section 6.

2. RELATED WORK

In this section, we describe related works. Table II lists the symbols used in our paper.

2.1. Triangle Counting

The global and local triangle counting in a graph has been extensively studied. The simplest and most time-efficient algorithm is to use matrix multiplication [Alon et al. 1997]—the diagonal elements of A^3 for an adjacency matrix A result in exact local triangle counting. Despite the fast running time of $O(m^{1.41})$, the algorithm is not suitable for large scale graphs due to its high space complexity of $O(n^2)$ where n and m are the numbers of nodes and edges, respectively. To achieve a fast running time with a reasonable space requirement, various approaches have been proposed [Latapy 2008; Schank and Wagner 2005].

To handle large-scale graphs, researchers have been also interested in devising external and distributed algorithms. Chu and Cheng [2011] proposed a graph partitioning based algorithm, and Hu et al. [2013] developed an algorithm by iteratively loading a part of edges and counting triangles among them. Recently, Pagh and Silvestri [2014] achieved the currently minimum I/O complexity $O(m^{3/2}/(B\sqrt{M}))$ where M is the total space and B is a size of data transfer block. To make an algorithm more scalable, parallel computing has been also widely considered. Cohen [2009] proposed the first MAPREDUCE algorithm for triangle enumeration. Suri and Vassilvitskii [2011] proposed a graph partitioning based algorithm, which is further improved by Park and Chung [2013]. For more parallel algorithms, we refer to Arifuzzaman et al. [2013], Kim et al. [2014], Park et al. [2014], and Park et al. [2016].

2.2. Graph Stream Mining

There have been numerous studies on graph stream mining. We first present studies on triangle counting in a graph stream, and then those on other graph mining problems.

2.2.1. Triangle Counting. Recently, there have been numerous studies on triangle counting in a graph stream.

Global Triangle Counting. The first study was conducted theoretically by Bar-Yossef et al. [2002]. Afterwards, its space bound was improved by Jowhari and Ghodsi [2005] and Buriol et al. [2006] successively. Recently, Jha et al. [2015] proposed a single pass algorithm based on wedge sampling to estimate the number of triangles and the clustering coefficient from a graph stream. The algorithm takes $O(\sqrt{n})$ memory spaces with an additive error guarantee. Kane et al. [2012] devised a sketch based streaming algorithm to count subgraphs with a constant size, which allows edge deletion. This result was improved in the space complexity by Pavan et al. [2013]. Tsourakakis et al. [2009] presented a graph sparsification method by edge sampling for estimating the number of triangles in a graph, which requires one-pass to the graph and can be applied to a stream model. A more general edge sampling framework was proposed by Ahmed et al. [2014], which can be applied to estimating various graph statistics including the number of triangles.

Local Triangle Counting. Counting triangles in a graph stream for each node has been also studied. Becchetti et al. [2010] proposed a semi-streaming algorithm requiring $\log n$ passes, and Kutzkov and Pagh [2013] proposed a single pass algorithm ("KP" henceforth) based on node coloring [Pagh and Tsourakakis 2012]. However, they are limited in practice: Becchetti et al. [2010] requires multiple passes to a graph, and Kutzkov and Pagh [2013] requires to know the number of nodes in a graph stream in advance. In contrast, our proposed MASCOT, whose preliminary version appeared in Lim and Kang [2015], requires only one pass to the graph with only one parameter, the edge sampling probability, while achieving a better accuracy. All of the presented works deal with a *simple graph*, and to the best of our knowledge, our proposed MULTIBMASCOT and MULTIWMASCOT are the first algorithms for local triangle counting in a *multigraph* stream. Although local triangle counting for graph streams with edge deletion has been recently studied with a fixed memory usage [De Stefani et al. 2016], handling duplicated edges still remains open.

2.2.2. Other Problems. Other graph mining tasks for a stream model have been extensively studied as well. Balanced graph partitioning in a graph stream was initiated in Stanton and Kliot [2012] by suggesting several simple heuristics. Their method not only significantly outperforms a naïve hashing scheme, but also is comparable to the state-of-the-art method for some cases. Stanton [2014] theoretically showed the lower bound $o(n)$ of approximation ratio, and proposed two randomized greedy algorithms. We refer to Nishimura and Ugander [2013] and Tsourakakis et al. [2014] for more related studies.

Another interesting topic is online PAGERANK computation. Sarma et al. [2011] studied performing a random walk from a graph stream, and their method enables computing PAGERANK and conductance. Also PAGERANK computation in an evolving graph has been studied in Bahmani et al. [2010], Bahmani et al. [2012] and Desikan et al. [2005].

3. PROPOSED METHOD FOR SIMPLE GRAPH

In this section, we propose MASCOT, a memory-efficient and accurate algorithm to provide unbiased estimation of local triangle counts for every node in a simple graph stream. We first present two naïve algorithms MASCOT-C and MASCOT-A based on edge sampling. While MASCOT-A provides lower variance than MASCOT-C, it requires more memory for the same sampling probability. MASCOT achieves both memory-efficiency and small variance. Also MASCOT requires only one simple parameter of edge sampling probability and no prior knowledge on an input graph.

3.1. MASCOT-C

MASCOT-C (Memory-efficient Accurate Sampling for Counting Local Triangles with Constant Sampling) is based on DOULION [Tsourakakis et al. 2009], a method to estimate the number of triangles in a massive graph by sparsification. MASCOT-C samples every edge from a graph stream with a constant probability p while keeping local triangle estimation up-to-date for every node. For an efficient update, given a new edge (u, v), only the estimations for $c \in \mathcal{N}_{uv} \cup \{u, v\}$, where $\mathcal{N}_{uv}$ is the set of common neighbors of u and v, are updated. Note that triangle counts of the other nodes are not affected by (u, v). The dominant factor of the total cost per new edge is the computation of $\mathcal{N}_{uv}$ which takes $O(\min(d_u, d_v))$ time where d_u is the degree of u of a sampled graph. The whole procedure of MASCOT-C is shown in ALGORITHM 1. For every new edge e, MASCOT-C samples it with a user-provided probability p (Line 5), and updates local triangle estimations only if the sampling succeeds (Line 6 and 7) which means that the number of triangles can change.

MASCOT-C provides unbiased estimation as stated in the following lemma.

LEMMA 3.1. *Let $\triangle_u$ be the true number of local triangles of u, and τ_u^c be its estimation by* MASCOT-C. *For every $u \in V$,*

$$\mathbb{E}\left[\tau_u^c\right] = \triangle_u.$$

PROOF. Let δ_λ indicate whether the triangle λ is sampled or not. At any time, the expected value of τ_u^c becomes

$$\mathbb{E}\left[\tau_u^c\right] = \sum_{\lambda \in \mathcal{T}_u} \frac{1}{p^3} \mathbb{E}\left[\delta_\lambda\right],$$

where $\mathcal{T}_u$ is the set of triangles containing u. Since every edge is sampled independently with probability p, $\mathbb{E}\left[\delta_\lambda\right] = p^3$. Hence, we obtain $\mathbb{E}\left[\tau_u^c\right] = \triangle_u$. □

The following lemma analyzes variance of MASCOT-C.

LEMMA 3.2. *Let $\triangle_u$ be the true number of local triangles of u, and τ_u^c be its estimation by* MASCOT-C. *At any time, for every $u \in V$,*

$$\mathrm{Var}\left[\tau_u^c\right] = \frac{\triangle_u \left(1 - p^3\right) + r_u \left(p^2 - p^3\right)}{p^3},$$

where $r_u = \sum_{v \in \mathcal{N}_u} |\mathcal{N}_{uv}| \left(|\mathcal{N}_{uv}| - 1\right)$, $\mathcal{N}_u$ is the set of neighbors of u, and $\mathcal{N}_{uv} = |\mathcal{N}_u \cap \mathcal{N}_v|$.

ALGORITHM 1: MASCOT-C (Memory-efficient Accurate Sampling for Counting Local Triangles with Constant Sampling)

Input: Graph stream S, and edge sampling probability p.
Output: Estimation τ for local triangle counts.

$G \leftarrow (V, E)$ with $V = E = \emptyset$.
foreach *edge* $e = (u, v)$ *from* S **do**
  $ReadyNode(u)$.
  $ReadyNode(v)$.
  $x \leftarrow SampleEdge(e, p)$.
  if $x = 1$ **then**
    $CountTriangles(e, 1/p^3)$.
  end
end
Function $ReadyNode(u)$
  if $u \notin V$ **then** $V \leftarrow V \cup \{u\}$ and $\tau_u = 0$.
end
Function $SampleEdge((u, v), p) \rightarrow$ **int**
  $x \leftarrow Bernoulli(p)$.
  if $x = 1$ **then**
    $E \leftarrow E \cup \{(u, v)\}$.
  end
  return x.
end
Function $CountTriangles((u, v), s)$
  $\mathcal{N}_{uv} \leftarrow \mathcal{N}_u \cap \mathcal{N}_v$.
  foreach $c \in \mathcal{N}_{uv}$ **do**
    $\tau_c \leftarrow \tau_c + s$.
  end
  $\tau_u \leftarrow \tau_u + |\mathcal{N}_{uv}|s$.
  $\tau_v \leftarrow \tau_v + |\mathcal{N}_{uv}|s$.
end

PROOF. Let δ_λ indicate whether the triangle λ is sampled or not. At any time, the variance of τ_u^c becomes

$$\begin{aligned}\mathrm{Var}\left[\tau_u^c\right] &= \mathrm{Var}\left[\frac{1}{p^3}\sum_{\lambda\in\mathcal{T}_u}\delta_\lambda\right] = \frac{1}{p^6}\sum_{\lambda\in\mathcal{T}_u}\sum_{\gamma\in\mathcal{T}_u}\mathrm{Cov}\left[\delta_\lambda, \delta_\gamma\right] \\ &= \frac{1}{p^6}\left(\sum_{\lambda\in\mathcal{T}_u}\mathrm{Var}\left[\delta_\lambda\right] + \sum_{\lambda\in\mathcal{T}_u}\sum_{\substack{\gamma\in\mathcal{T}_u \\ \gamma\neq\lambda}}\mathrm{Cov}\left[\delta_\lambda, \delta_\gamma\right]\right)\end{aligned}$$

By definition, $\mathrm{Var}\left[\delta_\lambda\right] = p^3 - p^6$ for any triangle λ. If two triangles λ and γ do not share an edge, $\mathrm{Cov}\left[\lambda, \gamma\right] = 0$; if they share one edge, $\mathrm{Cov}\left[\lambda, \gamma\right] = p^5 - p^6$. The number of triangle pairs adjacent to u which share an edge is calculated by examining each edge of u. For each edge $e = (u, v)$, u has $\mathcal{N}_{uv}$ number of triangles sharing e. Since any two triangles share at most one edge, the number of triangle pairs adjacent to u which share an edge becomes

$$r_u = \sum_{v\in\mathcal{N}_u}\mathcal{N}_{uv}\left(\mathcal{N}_{uv} - 1\right).$$

Thus,

$$\sum_{\lambda \in \mathcal{T}_u} \sum_{\substack{\gamma \in \mathcal{T}_u \\ \gamma \neq \lambda}} \mathrm{Cov}\left[\delta_\lambda, \delta_\gamma\right] = r_u \left(p^5 - p^6\right).$$

Since $|\mathcal{T}_u| = \triangle_u$ by definition, the lemma is proved. □

We also analyze the running time of MASCOT-C on two aspects as follows: 1) processing time per edge and 2) total running time. A main time-consuming operation is to compute the set $\mathcal{N}_{uv} = \mathcal{N}_u \cap \mathcal{N}_v$ of common neighbors for two incident nodes of each new edge (u, v) from a graph stream. We consider a HashMap for maintaining a neighbor list.

LEMMA 3.3. *Let $e = (u, v)$ be an edge which arrives at time t, $G = (V, E)$ be a sampled graph which contains sampled edges and nodes until time t, and $d_t(k)$ be a degree of node $k \in V$ at t. The running time for processing the edge $e = (u, v)$ is*

$$O(\min(d_t(u), d_t(v)).$$

PROOF. A dominant factor of processing a new edge is the computation of $\mathcal{N}_{uv}$. Without loss of generality, assume $d_t(u) \leq d_t(v)$. To compute $\mathcal{N}_{uv}$, we check each element of $\mathcal{N}_u$. If $\mathcal{N}_v$ contains the element, add it to the set $\mathcal{N}_{uv}$. Thus, the running time for processing a new edge is $O(\min(d_t(u), d_t(v)))$. □

LEMMA 3.4. *Let $G = (V, E)$ be a sampled graph which contains sampled edges and nodes until time t, $d_l(k)$ be the degree of node $k \in V$ at time l and $d^* = \max_{(u,v)\in E}(min(d_t(u), d_t(v)))$. The total running time of* MASCOT-C *is $O(d^*|E|)$.*

PROOF. Let $e_l \in E$ be the edge which arrived at time $l \leq t$. Until the current time t, MASCOT-C needs to process all of the sampled edges in E. By Lemma 3.3, the total running time of MASCOT-C becomes

$$\begin{aligned}\sum_{e_l=(u,v)\in E} O(\min(d_l(u), d_l(v))) &\leq \sum_{E} O(\min(d_t(u), d_t(v))) \\ &\leq \sum_{E} O(d^*) \\ &= O(d^*|E|).\end{aligned}$$

Note that $d_l(k) \leq d_t(k)$ since every sampled edge e_l has arrived before the time t. □

3.2. MASCOT-A

MASCOT-A further improves MASCOT-C by decreasing the variance using a non-uniform sampling: if a new edge $e = (u, v)$ closes a triangle (i.e., the new edge e increases the number of triangles at least by 1), MASCOT-A unconditionally samples e; otherwise it samples e with the default probability p. In this approach, since each edge has a different sampling probability, MASCOT-A maintains the sampling probability $q_{xy} \in \{p, 1\}$ for every sampled edge (x, y) to give an appropriate weight for sampled triangles containing (x, y).

ALGORITHM 2: MASCOT-A (Memory-efficient Accurate Sampling for Counting Local Triangles with Adaptive Sampling)

Input: Graph stream S, and edge sampling probability p.
Output: Estimation τ for local triangle counts.

```
1  G ← (V, E) with V = E = ∅.
2  foreach edge e = (u, v) from S do
3  |  ReadyNode(u).
4  |  ReadyNode(v).
5  |  if e forms a triangle then
6  |  |  SampleEdgeA(e, 1).
7  |  |  CountTrianglesA(e).
8  |  else
9  |  |  SampleEdgeA(e, p).
10 |  end
11 end
12 Function SampleEdgeA((u, v), p)
13 |  x ← Bernoulli(p).
14 |  if x = 1 then
15 |  |  E ← E ∪ {(u, v)}.
16 |  |  q_uv = p.
17 |  end
18 end
19 Function CountTrianglesA((u, v))
20 |  N_uv ← N(u) ∩ N(v).
21 |  foreach c ∈ N_uv do
22 |  |  s ← 1/(q_uv q_uc q_vc).
23 |  |  τ_c ← τ_c + s.
24 |  |  τ_u ← τ_u + s.
25 |  |  τ_v ← τ_v + s.
26 |  end
27 end
```

We first consider the former case, i.e., e forms a triangle. Let $\mathcal{N}_{uv} = \mathcal{N}_u \cap \mathcal{N}_v$ be a set of nodes constructing triangles $\{(u, v, c) \mid c \in \mathcal{N}_{uv}\}$ where $\mathcal{N}_u$ is a set of neighbors of u. By the non-uniform sampling, MASCOT-A increases the triangle counts of the nodes u, v and every $c \in \mathcal{N}_{uv}$. Precisely, for each new triangle (u, v, c), each estimated count for u, v and c is incremented by $1/q_{uv}q_{uc}q_{vc}$; note that in this case, $q_{uv} = 1$.

If the new edge $e = (u, v)$ forms no triangle, it is sampled with the default probability p, and if picked, q_{uv} is set to p. Regardless of whether the sampling for e succeeds or not, there is no estimation update since no triangle is newly formed. MASCOT-A is fully described in ALGORITHM 2 where $ReadyNode$ is the same as that in MASCOT-C. Like MASCOT-C, MASCOT-A provides unbiased estimation.

LEMMA 3.5. *Let $\triangle_u$ be the true number of local triangles of u, and τ_u^a be its estimation by* MASCOT-A. *At any time, for every $u \in V$,*

$$\mathbb{E}\left[\tau_u^a\right] = \triangle_u.$$

PROOF. Let δ_λ indicate whether the triangle λ is sampled or not. At any time, the expected value of τ_u^a becomes

$$\mathbb{E}\left[\tau_u^a\right] = \sum_{\lambda=(e,f,g)\in\mathcal{T}_u} \frac{1}{q_e q_f q_g} \mathbb{E}\left[\delta_\lambda\right],$$

where $\mathcal{T}_u$ is the set of triangles containing u. Since q_e is the probability that the edge e is sampled, the probability of sampling λ becomes $q_e q_f q_g$, leading to $\mathbb{E}[\delta_\lambda] = q_e q_f q_g$. Consequently, we obtain $\mathbb{E}[\tau_u^a] = |\mathcal{T}_u| = \triangle_u$. □

For MASCOT-A, we analyze an upper bound of variance which is smaller than the variance of MASCOT-C in Lemma 3.2.

LEMMA 3.6. *Let $\triangle_u$ be the true number of local triangles of u, and τ_u^a be its estimation by* MASCOT-A. *At any time, for every $u \in V$,*

$$\mathrm{Var}[\tau_u^a] \leq \frac{\triangle_u \left(1-p^2\right) + r_u \left(p-p^2\right)}{p^2}.$$

PROOF. In this proof, $q_\lambda = q_{efg} = q_e q_f q_g$ for a triangle $\lambda = (e,f,g)$. Let δ_λ indicate whether the triangle λ is sampled or not. At any time, the expected value of τ_u^a becomes

$$\begin{aligned}\mathrm{Var}[\tau_u^a] &= \mathrm{Var}\left[\sum_{\lambda\in\mathcal{T}_u} \frac{1}{q_\lambda}\delta_\lambda\right] = \sum_{\lambda\in\mathcal{T}_u}\sum_{\gamma\in\mathcal{T}_u} \mathrm{Cov}\left[\frac{\delta_\lambda}{q_\lambda}, \frac{\delta_\gamma}{q_\gamma}\right] \\ &= \sum_{\lambda\in\mathcal{T}_u} \frac{1}{q_\lambda^2}\mathrm{Var}[\delta_\lambda] + \sum_{\lambda\in\mathcal{T}_u}\sum_{\substack{\gamma\in\mathcal{T}_u \\ \lambda\neq\gamma}} \frac{1}{q_\lambda q_\gamma}\mathrm{Cov}[\delta_\lambda, \delta_\gamma].\end{aligned} \tag{1}$$

Note that for any sampled triangle λ, $q_\lambda \geq p^2$ by construction of the algorithm. Thus, the first term of Eq. (1) is bounded as follows:

$$\frac{1}{q_\lambda^2}\mathrm{Var}[\delta_\lambda] = \frac{1}{q_\lambda^2}\left(q_\lambda - q_\lambda^2\right) \leq \frac{1}{p^2}\left(1-p^2\right). \tag{2}$$

The remaining task is to bound the following in the second term:

$$\begin{aligned}\frac{1}{q_\lambda q_\gamma}\mathrm{Cov}[\delta_\lambda, \delta_\gamma] &= \frac{1}{q_\lambda q_\gamma}\left(\mathbb{E}[\delta_\lambda\delta_\gamma] - \mathbb{E}[\delta_\lambda]\,\mathbb{E}[\delta_\gamma]\right) \\ &= \frac{\mathbb{E}[\delta_\lambda\delta_\gamma]}{q_\lambda q_\gamma} - 1.\end{aligned}$$

Since $\mathrm{Cov}[\delta_\lambda, \delta_\gamma] = 0$ for independent triangles λ and γ, we focus on the case where $\lambda = (e,f,g)$ and $\gamma = (g,h,\ell)$ share the one edge g. There are two cases:

— g is the last among all the edges of λ and γ: In this case, $q_g = 1$; then the following holds:

$$\frac{\mathbb{E}[\delta_\lambda\delta_\gamma]}{q_\lambda q_\gamma} = \frac{q_e q_f q_h q_\ell}{q_{efg} q_{gh\ell}} = \frac{1}{q_g^2} = 1.$$

— g is not the last among all the edges of λ and γ: There are at least two edges ($\neq g$) that are unconditionally sampled. Let them be e and ℓ without loss of generality, i.e. $q_e = q_\ell = 1$; then

$$\frac{\mathbb{E}[\delta_\lambda\delta_\gamma]}{q_\lambda q_\gamma} = \frac{q_f q_g q_h}{q_{efg} q_{gh\ell}} = \frac{1}{q_e q_g q_\ell} \leq \frac{1}{p}. \tag{3}$$

Substituting Eq. (2) and (3) to Eq. (1), we prove the lemma. □

Note that both $\triangle_u$ and r_u are graph characteristics independent of the algorithms. As a result, MASCOT-A has a lower variance than MASCOT-C. The main drawback of MASCOT-A is that the number of edges sampled depends on the order of edges in

a stream. It makes hard to exactly analyze the space requirement of MASCOT-A: the size of sampled edges may vary highly between pm and m.

LEMMA 3.7. *Let $G = (V, E)$ be a sampled graph which contains sampled edges and nodes until time t, $d_l(k)$ be the degree of node $k \in V$ at time l and $d^* = \max_{(u,v)\in E}(min(d_t(u), d_t(v)))$. The total running time of* MASCOT-A *is $O(d^*|E|)$.*

PROOF. The lemma is proved in the same way as in Lemma 3.4. □

3.3. MASCOT

Although MASCOT-A has lower variance than MASCOT-C, it may sample too many edges. By the following observation, however, we do not need to unconditionally sample an edge even though it constructs a triangle.

OBSERVATION 1. *For a new edge $e = (u, v)$, let c be a node having edges to both u and v. In the future, e will not be used for constructing a triangle involving c.*

Namely, only u and v need e. Thus, c can safely discard e after counting the triangle, and u and v sample e with the given probability p for the future use. In other words, we do not need to sample e with an especially high probability, e.g. 1 in MASCOT-A. Following this idea, MASCOT counts all triangles newly formed by every incoming edge e unconditionally, and then samples e with the default probability p. ALGORITHM 3 shows the pseudocode of this process where $ReadyNode$, $CountTriangles$ and $SampleEdge$ are the same as those in MASCOT-C.

ALGORITHM 3: MASCOT (Memory-efficient Accurate Sampling for Counting Local Triangles)

Input: Graph stream S, and edge sampling probability p.
Output: Estimation τ for local triangle counts.

```
1 G ← (V, E) with V = E = ∅.
2 foreach edge e = (u, v) from S do
3 |   ReadyNode(u).
4 |   ReadyNode(v).
5 |   CountTriangles(e, 1/p^2).
6 |   SampleEdge(e, p).
7 end
```

MASCOT provides unbiased estimation as follows.

LEMMA 3.8. *Let $\triangle_u$ be the true number of local triangles of u, and τ_u be its estimation by* MASCOT. *At any time, for every $u \in V$,*

$$\mathbb{E}[\tau_u] = \triangle_u.$$

PROOF. Every new incoming edge remains in our sampled graph in the future with probability p. We show that every triangle $\lambda = (e, f, g)$ in the graph is counted with probability p^2. Without loss of generality, assume that e, f and g are given in order from the graph stream. Let δ_λ be an indicator representing whether λ is counted or not. For λ to be sampled, e and f should be sampled before g whose probability is p^2. In that case, when g is observed, λ is unconditionally counted. Thus, $\mathbb{E}[\delta_\lambda] = p^2$. With the weight of $1/p^2$ for sampled edges, the lemma is proved by following the proof of Lemma 3.5. □

LEMMA 3.9. *Let $\triangle_u$ be the true number of local triangles of u, and τ_u be its estimation by* MASCOT. *At any time, for every $u \in V$,*

$$\mathrm{Var}\left[\tau_u\right] \leq \frac{\triangle_u \left(1-p^2\right) + r_u \left(p-p^2\right)}{p^2}.$$

PROOF. With $\mathbb{E}\left[\delta_\lambda\right] = p^2$, the proof is almost the same as that for Lemma 3.2. The only difference is to compute $\mathrm{Cov}\left[\delta_\lambda, \delta_\gamma\right]$. In MASCOT, $\mathbb{E}\left[\delta_\lambda \delta_\gamma\right]$ for two triangles $\lambda, \gamma \in \mathcal{T}_u$ sharing one edge varies depending on the order of edges in a graph stream. If the shared edge of λ and γ is the last or the second last among all edges of λ and γ, $\mathbb{E}\left[\delta_\lambda \delta_\gamma\right] = p^4$; otherwise $\mathbb{E}\left[\delta_\lambda \delta_\gamma\right] = p^3$. Thus, $\mathbb{E}\left[\delta_\lambda \delta_\gamma\right] \leq p^3$, which proves the lemma. □

While the upper bound of variance of MASCOT is the same as that of MASCOT-A, MASCOT samples a smaller number of edges than MASCOT-A, which means that it requires smaller memory spaces for the same p. Moreover, the number of sampled edges of MASCOT is easily estimated as pm where m is the number of edges occurring in the stream until currently.

Running time of MASCOT is different from those of MASCOT-C and MASCOT-A. MASCOT updates the estimation before sampling an edge. Then, it computes the set $\mathcal{N}_{uv}$ not $|E|$ times but m times, where $|E|$ and m are the number of the sampled and the whole graph edges, respectively.

LEMMA 3.10. *Let $G = (V, E)$ be a sampled graph which contains sampled edges and nodes until time t, $G' = (V', E')$ be the whole graph, $d_l(k)$ be the degree of node $k \in V$ at time l in G and $d^* = \max_{(u,v)\in E'}(min(d_t(u), d_t(v)))$. The total running time of* MASCOT *is $O(d^*|E'|)$.*

PROOF. Let $e_l \in E$ be the edge which arrived at time $l \leq t$. Until time t, MASCOT needs to process all of edges in E'. Then, the total running time of MASCOT becomes

$$\begin{aligned}\sum_{e_l=(u,v)\in E'} O(\min(d_l(u), d_l(v))) &\leq \sum_{E'} O(\min(d_t(u), d_t(v))) \\ &\leq \sum_{E'} O(d^*) \\ &= O(d^*|E'|).\end{aligned}$$

□

4. PROPOSED METHOD FOR MULTIGRAPH

In this section, we present two algorithms for local triangle estimation from multigraph stream, a graph stream with duplicated edges. Simply applying the approach in Section 3 does not work for a multigraph stream because the sampling probability of each edge is unknown. For example, let an edge e occur c times in the stream; then the probability of sampling e is equal to $1-(1-p)^c$. But this requires recording the number of occurrences for every edge regardless of the result of the sampling, which means that we need to keep all distinct edges.

Below, we describe two algorithms: MULTIBMASCOT for unweighted counting and MULTIWMASCOT for weighted counting. For the unweighted counting, we consider if there is a triangle or not for each node triple, while for the weighted counting, each triangle is weighted depending on the number of occurrences of the participating edges.

4.1. MultiBMASCOT

MULTIBMASCOT is designed for binary triangle counting in a multigraph stream. This corresponds to ignoring the effect of duplicated edges, and the result by MULTIBMASCOT on a multigraph becomes the same as that by MASCOT-C on the corresponding simple graph. MULTIBMASCOT is especially desired when considering a multigraph stream as its corresponding simple graph stream without explicit conversion which can be only applied at the end of the stream.

ALGORITHM 4: MULTIBMASCOT (Unweighted Local Triangle Counting for Multigraph Stream)

Input: Graph stream S, and edge sampling probability p.
Output: Estimation τ for local triangle counts.

$G \leftarrow (V, E)$ with $V = E = \emptyset$.
foreach *edge* $e = (u, v)$ *from* S **do**
  $ReadyNode(u)$.
  $ReadyNode(v)$.
  $ProcessEdge(e, p)$.
end
Function $ProcessEdge((u, v), p)$
  $x \leftarrow Bernoulli(p)$.
  if $x = 1$ **then**
    if $(u, v) \notin E$ **then**
      $E \leftarrow E \cup \{(u, v)\}$.
      $CountTriangles(e, 1/p^3)$.
    end
  else
    if $(u, v) \in E$ **then**
      $E \leftarrow E \setminus \{(u, v)\}$.
      $N_{uv} \leftarrow \mathcal{N}(u) \cap \mathcal{N}(v)$.
      foreach $c \in \mathcal{N}_{uv}$ **do**
        $\tau_c \leftarrow \tau_c - 1/p^3$.
      end
      $\tau_u \leftarrow \tau_u - |N_{uv}|/p^3$.
      $\tau_v \leftarrow \tau_v - |N_{uv}|/p^3$.
    end
  end
end

The main idea is that sampling of each edge is determined only by the sampling of its last occurrence. In other words, every occurrence of e results in adding e to and removing e from the sampled graph with probability p and $1-p$, respectively. As a result, regardless of the amount of edge duplication, the probability of sampling each edge is equal to p at any time. The difference of MULTIBMASCOT from MASCOT-C is that in MULTIBMASCOT any edge already sampled can be removed from the sampled graph. Thus, we need to handle the case of the edge removal in triangle estimation, which corresponds to a reverse operation of $CountTriangles$ in ALGORITHM 1. MULTIBMASCOT is fully described in ALGORITHM 4.

LEMMA 4.1. MULTIBMASCOT *provides unbiased estimation for every node.*

PROOF. The lemma is proved in the same way as Lemma 3.1 by the fact that the probability of sampling each edge is p regardless of the degree of its duplication. □

Note that MULTIBMASCOT on a multigraph stream is the same as MASCOT-C on the corresponding simple graph stream, but requires no graph conversion to remove duplicated edges.

4.2. MultiWMASCOT

MULTIWMASCOT is an algorithm for weighted counting of local triangles in a multigraph stream. Let h_e be the number of repeated occurrences of an edge e[1]; then we consider h_e as the weight of e in the graph. This weighting has various real world applications, including social, email, router, collaboration and co-purchasing networks.

Given an edge weight h, in this paper, we especially focus on the weighting function $f(h_{uv}, h_{uw}, h_{vw}) = h_{uv}h_{uw}h_{vw}$ for each triangle $\lambda = (h_{uv}, h_{uw}, h_{vw})$. Note that if any of h_{uv}, h_{uw}, and h_{vw} is equal to 0, the corresponding weight should be 0, i.e. $f(h_{uv}, h_{uw}, h_{vw}) = 0$.

For the purpose, we propose MULTIWMASCOT for weighted triangle counting in multigraph streams. MULTIWMASCOT is also based on edge sampling, and maintains the number θ_e of occurrences in the sampled graph for every edge e. The whole procedure is described in ALGORITHM 5. In contrast to the binary counting, we are able to apply the approach of MASCOT for weighted counting. That is, whenever a new edge $e = (u, v)$ arrives, local triangle estimation is unconditionally updated for u, v, and $\mathcal{N} = \mathcal{N}_u \cap \mathcal{N}_v$. For every new triangle $\lambda = (u, v, c)$ where $c \in \mathcal{N}$, the expected increase of the weight of the triangle λ is $\theta_{cu}\theta_{cv}/p^2$ (Line 17 of ALGORITHM 5). Note that θ_{cu}/p is the expected number of edge occurrences between c and u, and the increase of expected edge occurrences between u and v is 1. After the update, the algorithm determines whether to keep e or not by sampling with probability p.

LEMMA 4.2. MULTIWMASCOT *provides unbiased estimation for every node.*

PROOF. Let us consider MASCOT, which independently counts triangles consisting of the same node triple with duplicated edges. That is, for $e_1 = e_2 = \{u, v\}$, $e_3 = e_4 = \{v, w\}$, and $e_5 = \{w, u\}$, the four triangles by $\{e_1, e_2\} \times \{e_3, e_4\} \times \{e_5\}$ are considered independently whenever each of them is formed. Then, the lemma is proved in the same way as Lemma 3.8. □

5. EXPERIMENTS

In this section, we show experimental results to answer the following questions.

- Q1 How accurate is MASCOT compared to its naïve versions? (Section 5.3)
- Q2 What is the performance of MASCOT compared with the previous local triangle counting methods [Kutzkov and Pagh 2013; Becchetti et al. 2008]? (Section 5.4)
- Q3 How accurate are MULTIBMASCOT and MULTIWMASCOT? (Section 5.5)
- Q4 What is the performance of MASCOT compared with the previous global triangle counting methods [Jha et al. 2015; Pavan et al. 2013]? (Section 5.6)
- Q5 What are the discoveries from the proposed methods? (Section 5.7)

5.1. Datasets

We gather graph data from diverse domains such as social networks, router connectivity, hyperlinks in webpages, collaboration networks, citation networks, etc. We make them undirected and unweighted with no self-loop. Their edges are given in a random order. Table III lists the datasets used in our experiments.

[1]We also use h_{uv} to denote h_e for $e = (u, v)$.

ALGORITHM 5: MULTIWMASCOT (Weighted Local Triangle Counting for Multigraph Stream)

Input: Graph stream S, and edge sampling probability p.
Output: Estimation τ for local triangle counts.

$G \leftarrow (V, E)$ with $V = E = \emptyset$.
foreach *edge* $e = (u, v)$ *from* S **do**
  $ReadyNode(u)$.
  $ReadyNode(v)$.
  $CountWeightedTriangles(e)$.
  $SampleWeightedEdge(e)$.
end
Function $SampleWeightedEdge(e)$
  if $Bernoulli(p) = 1$ **then**
    if $e \in E$ **then** $\theta_e = \theta_e + 1$.
    else $E \leftarrow E \cup \{e\}$; $\theta_e = 1$.
  end
end
Function $CountWeightedTriangles((u, v))$
  $\mathcal{N}_{uv} \leftarrow \mathcal{N}_u \cap \mathcal{N}_v$.
  foreach $c \in \mathcal{N}_{uv}$ **do**
    $\omega = \theta_{cu}\theta_{cv}/p^2$.
    $\tau_c \leftarrow \tau_c + \omega$.
    $\tau_u \leftarrow \tau_u + \omega$.
    $\tau_v \leftarrow \tau_v + \omega$.
  end
end

5.2. Evaluation Metrics

To evaluate local triangle counting algorithms, we consider the following metrics.

— Pearson Correlation Coefficient ρ: This measures how well the relationship between two variables x and y is represented by a linear function. Given two vectors x and y, the definition is as follows:

$$\rho(x, y) = \frac{\mathrm{Cov}\,[x, y]}{\sigma_x \sigma_y}.$$

— Mean ϵ of Error: This measures how close our estimation is to the ground truth. Given an estimation x for the ground truth x^*, we use the following absolute relative error:

$$\epsilon(x, x^*) = \frac{1}{n} \sum_{i=1}^{n} z_i,$$

where $z_i = |x_i - x_i^*| / (x_i^* + 1)$. We add 1 to both x and x^* for the case that $x_i^* = 0$.

— This measures how large is the error of our estimation to the ground truth relatively. Given an estimation x for the ground truth x^*, the definition is as follows:

$$\psi = \frac{|x - x^*|}{x^*}$$

— Ratio η of Sampled Edges: This dominates the amount of memory spaces required by an algorithm. It is equal to p for MASCOT, MASCOT-C and MULTIBMASCOT, and larger than p for MASCOT-A and MULTIWMASCOT in expectation.

Table III. Summary of the graph data used in our experiments. The number of nodes and edges are counted after removing direction, weights, and self-loops.

Name	Nodes	Edges	Description
Simple Graphs			
Advogato[1]	5,155	39,285	Trust network
Enron[2]	36,692	183,831	Enron email exchanges
Wiki-Conflict[1]	116,836	2,027,871	Edit conflict
Gowalla[2]	196,591	950,327	Online social network
MovieRev9[2]	253,045	6,611,899	Co-reviewed movies in Amazon
Stanford[2]	281,903	1,992,636	Web graph of Stanford.edu
NotreDame[2]	325,729	1,090,108	Web graph of Notre Dame
Petster[1]	623,766	15,699,276	Social websites for cat and dog owners
BerkStan[2]	685,230	6,649,470	Web graph of Berkeley and Stanford
DBLP-S[1]	1,314,050	5,362,414	Co-author network in DBLP
LiveJournal[2]	4,846,609	42,851,237	LiveJournal online social network
Multigraphs			
Actor[1]	382,219	33,115,812	Actor collaboration in movies
Baidu[1]	415,641	3,284,387	"related to" links in Baidu Encyclopedia
DBLP-M[1]	1,314,050	18,986,618	Co-author network in DBLP
ItWiki[1]	1,703,605	86,548,398	Hyperlinks in Italian Wikipedia
ChinWiki[1]	1,930,275	9,359,108	Hyperlinks in Chinese Wikipedia
PhoneCall	26,578,926	480,652,650	Phone call history

[1]http://konect.uni-koblenz.de/
[2]http://snap.stanford.edu/data/index.html

Note that the first two metrics are calculated for the true and estimated local triangle counts.

We use the average of measurements obtained by 10 independent runnings since our algorithms are randomized. For the competing method KP [Kutzkov and Pagh 2013], the same averaging scheme is used since the implementation is based on random hashing.

5.3. Performance of MASCOT Compared to The Naïve Versions

Figure 2 shows Pearson correlation coefficients (PCC) over ratios of the number of sampled edges for our proposed algorithms. In general, all algorithms improve PCC as the sampling rate gets larger. Note that while the sampling rates of MASCOT-C and MASCOT are determined by p in expectation, that of MASCOT-A depends on both p and the edge order in a graph stream. For all the graphs, MASCOT and MASCOT-A show higher correlations than MASCOT-C at the same number of sampled edges. The difference between MASCOT and MASCOT-A is insignificant.

Figure 13 shows the mean ϵ of absolute relative error over the ratio η of sampled edges. Note that we also present the standard deviation of ϵ obtained by 10 runnings,

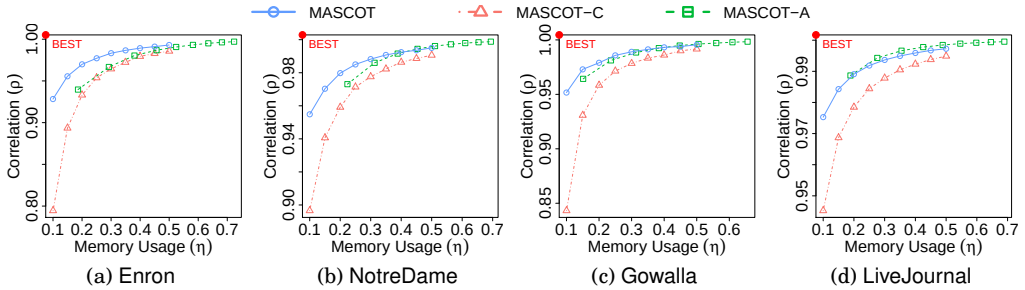


Fig. 2. Pearson correlation coefficient ρ over different ratio η of sampled edges, which dominate the required memory spaces. MASCOT shows the best performance.

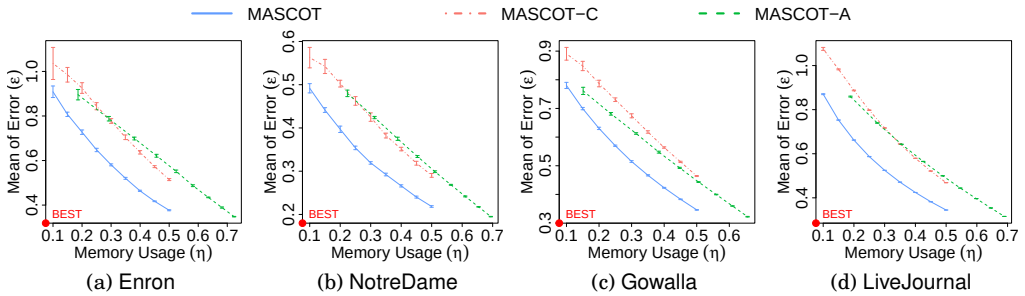


Fig. 3. Mean ϵ of absolute relative error over ratio η of the number of sampled edges of MASCOT and the basic versions of MASCOT. For the same η, MASCOT always results in the lowest error. For all graphs, as expected, standard deviations of MASCOT and MASCOT-A are smaller than that of MASCOT-C—notable for large graphs like LiveJournal. Sometimes, MASCOT-C is more accurate than or at least comparable to MASCOT-A. One reason is that for the same number of the total sampled edges, MASCOT-A samples large degree nodes more than MASCOT-C. This results in sacrificing accuracy for many nodes with extremely small degrees. The result with excluding nodes whose degrees are smaller than $2/p$ is shown in Figure 4.

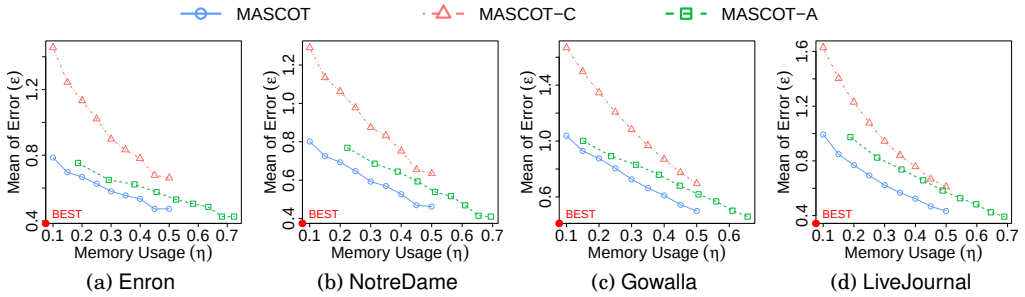
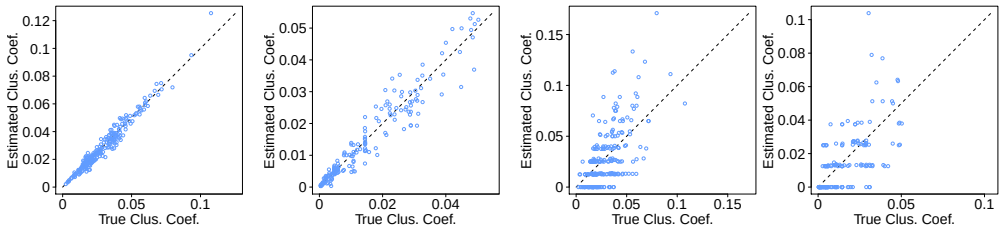


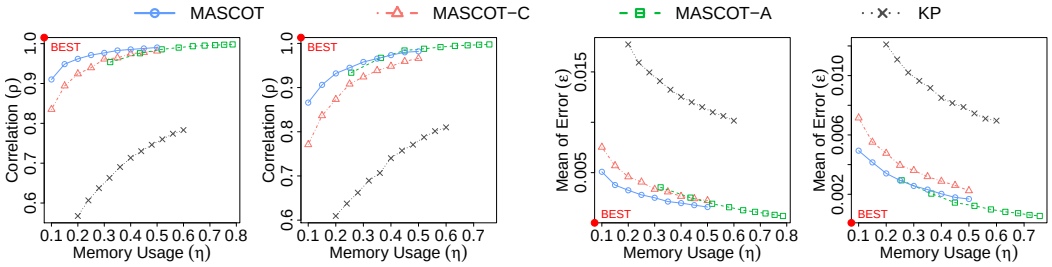
Fig. 4. Mean ϵ of relative error for nodes with degrees larger than $2/p$ over ratio η of the number of sampled edges of MASCOT and its variants. For the same η, sampled edges of MASCOT-A are more concentrated on nodes having many triangles than those of MASCOT-C are; for those nodes the error of MASCOT-A is smaller than that of MASCOT-C.

as stated in Section 5.2, for every point. For all graphs, including those not shown in the figure, MASCOT works the best under the same sampling rate. As shown in Section 3, standard deviations of MASCOT and MASCOT-A are smaller than MASCOT-C—especially remarkable when memory usage (η) is small.



(a) Wiki-Conflict by MASCOT (b) Stanford by MASCOT (c) Wiki-Conflict by KP (d) Stanford by KP

Fig. 5. The true clustering coefficient vs. its estimation by MASCOT with $p = 0.3$—(a) and (b)—and by KP with $K = 80$—(c) and (d). In this setting, both algorithms sample similar numbers of edges: $0.3m$ and $0.32m$ in expectation for MASCOT and KP, respectively. All plots are with respect to nodes having degrees at least 1000. Note that MASCOT estimates local triangles more accurately than KP—the points of MASCOT are nearly on the $y = x$ line in contrast to those of KP.



(a) Wiki-Conflict (b) Stanford (c) Wiki-Conflict (d) Stanford

Fig. 6. Pearson correlation coefficient ρ and mean ϵ of absolute relative error for the nodes with degrees at least 1000. While MASCOT is the best, MASCOT-C and MASCOT-A also outperform KP in terms of both metrics ρ and ϵ.

In contrast to the PCC case, the mean error of MASCOT-C and MASCOT-A varies depending on graphs. For example, MASCOT-C outperforms MASCOT-A for Enron and NotreDame in general, while MASCOT-A outperforms MASCOT-C for Gowalla; for LiveJournal they are comparable. The reason of this result is as follows. The triangle estimation is inaccurate for nodes with degrees smaller than $2/p$. This is because if a node has a degree less than $2/p$, the number of sampled incident edges of the node is less than 2 in expectation, leading to no chance to count its local triangles. For the same η, the edge sampling probability p_c for MASCOT-C is larger than the probability p_a for MASCOT-A, i.e. $2/p_c \leq 2/p_a$. As a result, MASCOT-A becomes accurate for a small number of large degree nodes, and not for a large number of small degree nodes, leading to large errors in total. This is shown in Figure 4 where the error ϵ is calculated for nodes whose degrees are above $2/p$. Note that MASCOT-A always outperforms MASCOT-C.

5.4. Comparison of MASCOT with Competing Local Triangle Counting Methods

We compare the accuracy and the running time of MASCOT with those of competing methods for local triangle counting. The experiments are performed on a server with 3.5GHz Intel CPUs and 32GB memory, and the methods are implemented in Java 1.8.0.

We consider the first and the only local triangle counting algorithm KP for a graph stream, proposed in Kutzkov and Pagh [2013]. KP generates multiple independent

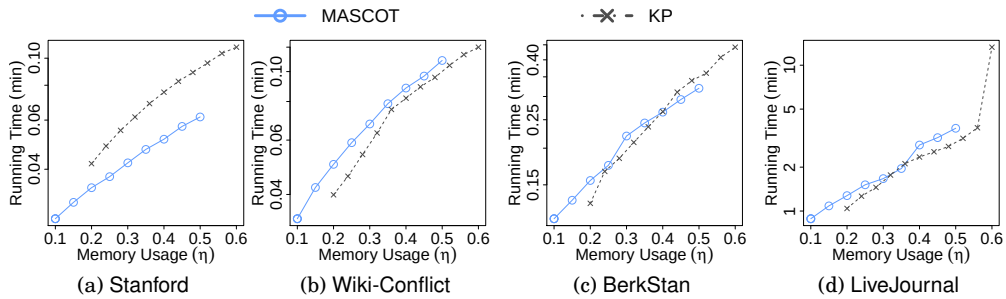


Fig. 7. Running time over a ratio η of the number of sampled edges for MASCOT and KP. Note that the running time of MASCOT is close to that of KP on average. In the Stanford, MASCOT is faster than KP. In the Wiki-Conflict, KP is faster than MASCOT.

sparsified graphs, and aggregates triangles from them. Since the method requires determining hash functions to map every node to a color before running, we assume that the number of nodes is known in advance. Following their experiments, we set $d = 1000$, that is, our focus is on nodes whose degrees are at least d. The other parameters are set according to Theorem 2 in their paper—the number C of colors to $d/4 = 250$ and the sparsification threshold t to $9m/d = 0.009m$—while the number K of independent sessions sparsifying an input graph is varied from 50 to 150 at the interval of 10. Note that the number of edges sampled becomes $\approx mK/C$. To compare the accuracy of MASCOT with KP, we measure the estimation accuracy of local clustering coefficients of nodes with degrees larger than 1000. Note that KP was primarily developed for local clustering coefficient estimation, and MASCOT provides it by dividing τ_u by $d_u(d_u-1)/2$ for each node u since τ is an unbiased estimator[2]. Calculating the degree of each node in a graph stream is trivial, i.e. counting for each node its occurrences in the stream.

Additionally, we compare MASCOT with the *semi-streaming* algorithm (BA) proposed in Becchetti et al. [2010] which requires *multiple passes* to a graph. We set the number P of passes to 10, 50, and 100. BA needs to set the number of random bits for a semi-permutation since the method allocates a random value to each node. Following their experiments, we set the number of bits to $\log(n)$. We do not consider ratios η of sampled edges of MASCOT and BA since a memory usage of BA does not depend on m: the space complexity of BA is $O(n)$. We set p of MASCOT to $0.3, 0.5, 0.7$.

Figures 1b, 1c and 5 show how well two methods MASCOT and KP estimate local clustering coefficients for the top-1000 high degree nodes. Note that all points for MASCOT are nearly on the $y = x$ line while KP's estimations are not that accurate despite somewhat positive correlations. Figure 6 shows comparison results of MASCOT and KP with respect to ρ and ϵ. For both graphs, MASCOT shows a high Pearson correlation coefficient ρ even with a small ratio η of sampled edges while KP's PCCs are below 0.8 regardless of the number of sampled edges. Furthermore MASCOT outperforms in the absolute relative error. For the other graphs used in Section 5.3, we obtain similar results.

Figure 7 shows running time of MASCOT and KP over memory usages η. Note that the running time of MASCOT is close to that of KP on average. In the Stanford, MASCOT is faster than KP. In the Wiki-Conflict, KP is faster than MASCOT.

[2]This is the same for MASCOT-C and MASCOT-A.

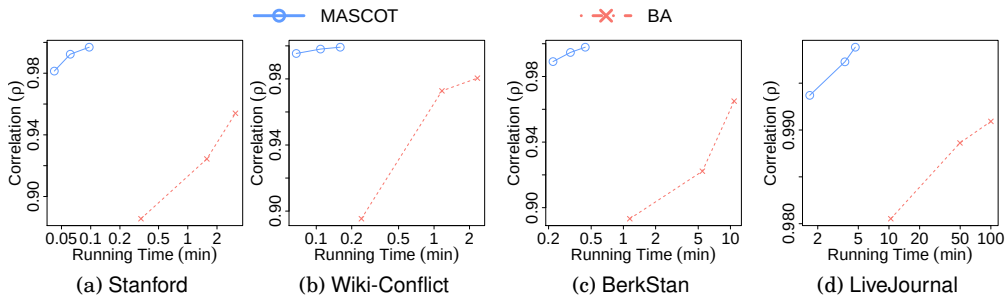


Fig. 8. Pearson correlation coefficient ρ over running time for MASCOT and BA. The pearson correlation coefficient of MASCOT is higher than that of BA. Also, the running time of MASCOT outperforms that of BA since BA requires multiple passes to a graph.

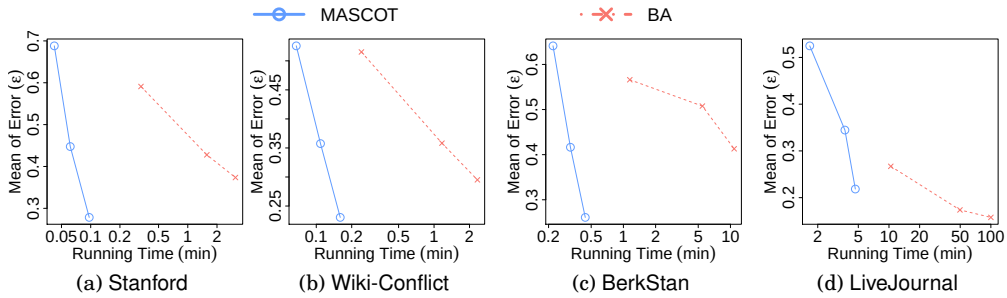


Fig. 9. Mean ϵ of absolute relative error over running time for MASCOT and BA. Note that in most graphs except LiveJournal, MASCOT has smaller minimum errors than BA does. Furthermore MASCOT runs much faster than BA for a given mean error.

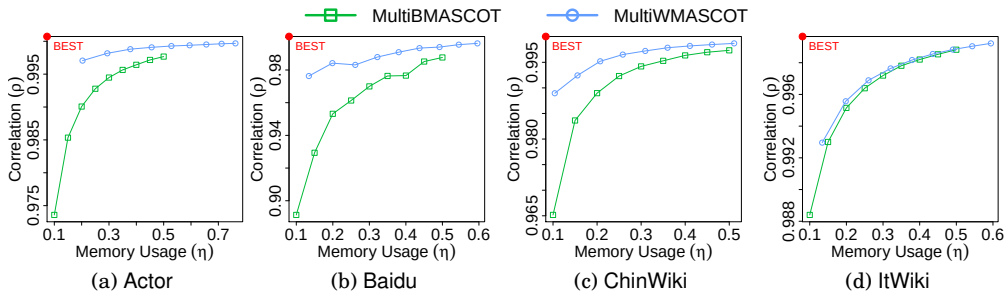


Fig. 10. Pearson correlation coefficient ρ over the ratio η of sampled edges for MULTIBMASCOT and MULTIWMASCOT. For all of graphs, MULTIBMASCOT and MULTIWMASCOT show high PCC: the performance becomes better as the memory usage increases.

Figures 8 shows pearson correlation coefficients ρ of MASCOT and BA over running time. The pearson correlation coefficient of MASCOT is higher than that of BA. Also, the running time of MASCOT outperforms that of BA since BA requires multiple passes to a graph.

Figures 9 shows means of error ϵ of MASCOT and BA over running time. Note that in most graphs except LiveJournal, MASCOT has smaller minimum errors than BA does. Furthermore MASCOT runs much faster than BA for a given mean error.

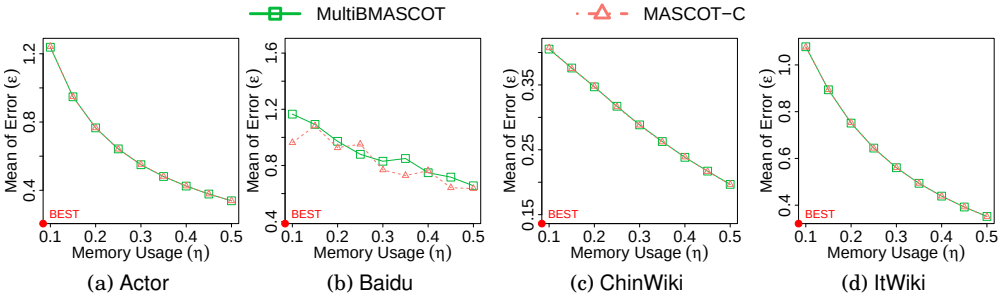


Fig. 11. Mean ϵ of absolute relative error over the ratio η of the number of sampled edges for MULTIBMASCOT, compared with MASCOT-C on the corresponding simple graph stream. MULTIBMASCOT performs as good as MASCOT-C in general. Note that MULTIBMASCOT is applied to a multigraph stream directly in contrast to MASCOT-C which requires graph conversion to remove duplicated edges. For Baidu, performance gap between MULTIBMASCOT and MASCOT-C is relatively large compared with the other data. This is because Baidu has a large number of small degree nodes with nonzero triangles for which error is relatively large.

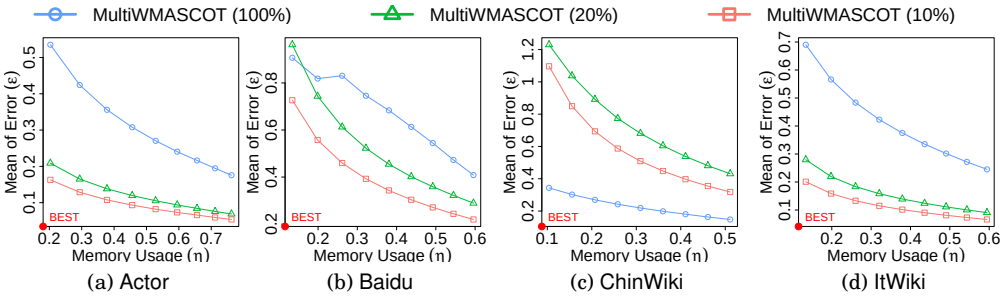


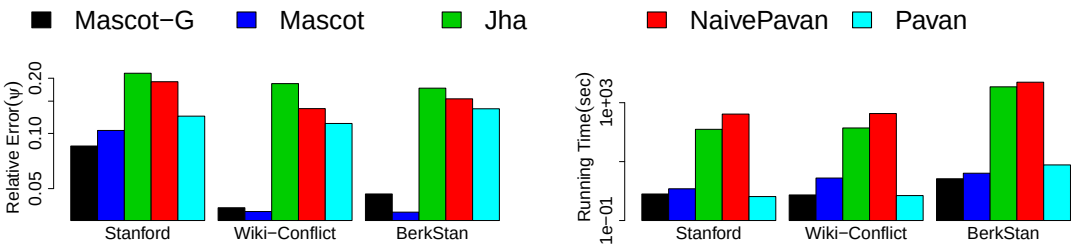
Fig. 12. Mean ϵ of absolute relative error over the ratio η of the number of sampled edges for MULTIWMASCOT. The red, green and blue lines correspond to results for top-100%, top-20% and top-10% high degree nodes, respectively. Generally, the estimation gets better for higher degree nodes. For ChinWiki, the pattern is reversed since a large portion of nodes, about 62%, have a degree less than 2, leading to no error for those nodes—estimation is always 0 which equals to the ground truth.

5.5. Performance of MultiBMASCOT and MultiWMASCOT

Figure 10 shows PCC ρ of MULTIBMASCOT and MULTIWMASCOT over the ratio η of the number of sampled edges. For all graphs, MULTIBMASCOT and MULTIWMASCOT show high correlations even with small memory usages. Note that the number of edges sampled by MULTIWMASCOT is larger than pm in general. This is because the chance of sampling an edge e in MULTIWMASCOT is larger than p, which is equal to $1-(1-p)^c$ where c is the number of occurrences of e, while the chance in MULTIBMASCOT is p.

Figure 11 shows the mean ϵ of absolute relative error of MULTIBMASCOT over the ratio η of sampled edges. MULTIBMASCOT performs as good as MASCOT-C. Note that MASCOT-C can run only after converting a multigraph stream to the corresponding simple graph while MULTIBMASCOT works on the multigraph stream directly. The performance of MULTIBMASCOT on Baidu slightly differs from that of MASCOT-C. This is because of a large number of small degree nodes with nonzero triangles in Baidu. Note that the estimations for these nodes have relatively large error. For Baidu, 53.7% of nodes have a degree less than 10 and at least one triangle while that proportion is 11.2% and 12% for Actor and ChinWiki, respectively.

Figure 12 shows the mean ϵ of absolute relative error of MULTIWMASCOT over the ratio η of sampled edges. Since there is no competitor, we present the performance of



(a) The Relative Error ψ (b) Running time (sec)

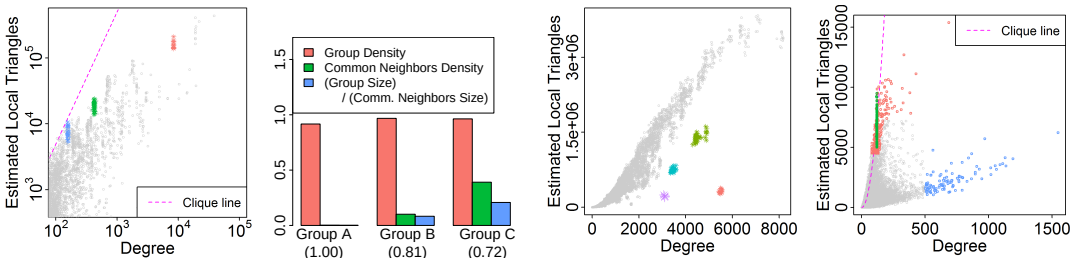
Fig. 13. (a) Relative error ψ of MASCOT compared to competing methods JHA, NAIVEPAVAN, and PAVAN. Note that our proposed methods outperform the other methods. The relative error of MASCOT-G is close to that of MASCOT. The relative error of MASCOT is 1.4x $\sim$ 3.6x times smaller than that of the second best method PAVAN. Note that the theoretical accuracy guarantees of PAVAN and NAIVEPAVAN are same. (b) Running time of the methods. our proposed methods outperform JHA and NAIVEPAVAN. Note that MASCOT-G is faster than MASCOT. The running time of JHA and NAIVEPAVAN are 156.8x $\sim$ 1899.2x times slower than that of MASCOT since time complexities of JHA and NAIVEPAVAN are $O(m^2)$. The running time of MASCOT is close to that of PAVAN. Note that PAVAN performs an efficient batch update and its time complexity is $O(m + r)$.

MULTIWMASCOT with regard to different ratios of high degree nodes. The red, green and blue lines indicate results for the top-100% (all nodes), top-20% and top-10% high degree nodes, respectively. In general, the estimation on the high degree nodes is more accurate than that on all nodes. This result is explained by the law of large numbers with a positive correlation between a degree and the number of triangles. Note that our estimation is the sum of random variables each of which corresponds to one triangle. This trend is, however, reversed in ChinWiki, i.e. estimation is more accurate for all nodes than high degree nodes. The reason is that a large number of nodes in ChinWiki, about 62% of the total, have a degree less than 2. Note that for those nodes, the error is always 0 since they participate in no triangle as well as being estimated to have no triangle. As a result, the error over all nodes becomes smaller than that over high degree nodes since error is always 0 for a large portion of nodes with extremely small degrees.

5.6. Performance of MASCOT in Global Triangle Counting

We compare accuracy and running time of our proposed methods MASCOT and MASCOT-G, a variant of MASCOT for global triangle counting, with those of three streaming based global triangle counting methods proposed in Jha et al. [2015] and Pavan et al. [2013] on Stanford, Wiki-Conflict, and BerkStan. MASCOT-G, based on MASCOT, stores only the global triangle count instead of each node's triangle count. Since MASCOT and MASCOT-G sample mp edges in expectation, we set the number of sampled edges of the competing methods to mp where m is the number of edges and p is a sampling probability set to 0.01. For the method proposed in Jha et al. [2015], which we call JHA, we set the sizes of edge and wedge pools to $mp/3$, respectively, since one wedge consists of two edges. There are two algorithms in Pavan et al. [2013]. The first method NAIVEPAVAN updates the estimation whenever a new edge arrives. The second method PAVAN brings better performance using bulk processing. Both NAIVEPAVAN and PAVAN use estimators each of which stores 2 edges; the number of estimators in both methods is set to $mp/2$. PAVAN needs an additional batch size parameter which is set to $4mp$ following their experiments.

Figure 13a shows relative errors ψ of MASCOT and MASCOT-G compared to competing methods JHA, NAIVEPAVAN, and PAVAN. Note that our proposed methods outperform the other methods. The relative error of MASCOT-G is close to that of MASCOT.



(a) Stanford (b) Stanford (c) MovieRev9 (d) DBLP-S

Fig. 14. Anomaly detection results for Stanford, Movie9 and DBLP-S. (a) Each color group forms a near clique, which is observed as a short vertical line in the scatter plot. The nodes in the group have a large number of common neighbors whose connections are sparse. The size of the common neighbors is much larger than that of the group such that the two groups form a core-periphery. (b) Bar graph showing the internal densities of each discovered group and their common neighbor group. Note that the discovered group has an edge density larger than 0.95 while the corresponding neighbor group is rarely connected. The value below each group means the ratio of common neighbors over all neighbors of the group members. (c) Each color group corresponds to a movie series which can be favored by people with diverse preferences: classic movies (green, blue and purple) and religious ones (red). (d) The steep linear pattern in red corresponds to researchers who participate in at least one paper with many coauthors: they are close to the clique line. The gradual linear pattern in blue corresponds to popular names. Since people with the same name becomes one point, the point covers various domains, leading to a large degree but weak local cohesiveness.

The relative error of MASCOT is 1.4x $\sim$ 3.6x times smaller than that of the second best method PAVAN. Note that the theoretical accuracy guarantees of PAVAN and NAIVEPAVAN are same.

Figure 13b shows running time of MASCOT and MASCOT-G compared to the competing methods. our proposed methods outperform JHA and NAIVEPAVAN. Note that MASCOT-G is faster than MASCOT. The running time of JHA and NAIVEPAVAN are 156.8x $\sim$ 1899.2x times slower than that of MASCOT since time complexities of JHA and NAIVEPAVAN are $O(m^2)$. Note that the time complexity of JHA is $O(ms_w)$ where $s_w = mp/3$ is the size of wedge pool and that of PAVAN is $O(mr)$. The running time of MASCOT is close to that of PAVAN. Note that PAVAN performs an efficient batch update and its time complexity is $O(m + r)$.

5.7. Anomaly Detection in Graph Streams

It has been known that local triangle counts play an important role in determining characteristics of nodes [Akoglu et al. 2010; Becchetti et al. 2010]. In this section, we show that our proposed algorithms detect anomalous nodes or patterns in real world graph streams. We use the Stanford, MovieRev9, DBLP-S, DBLP-M, PhoneCall, and Petster graphs listed in Table III.

5.7.1. Setup. We use MASCOT, MULTIBMASCOT and MULTIWMASCOT with $p = 0.3$ for all data. Stanford is a hyperlink network of webpages: a node and an edge correspond to a web page and a hyperlink, respectively. MovieRev9 is originally given as a list of movie reviews, containing the information of products and users, of Amazon. Each node of MovieRev9 corresponds to a product of the reviews; we make an undirected edge if two products have at least 9 common reviewers. DBLP is a collaboration network where there is an undirected edge between two authors whenever they participate in the same work. For DBLP, we use both the original multigraph (DBLP-M) and the corresponding simple graph (DBLP-S). Petster is a friendship network in social

websites catster.com and dogster.com for pet owners. PhoneCall is a phone call history in which each record consists of a sender, a receiver and time.

For Stanford and PhoneCall, we focus only on structural anomalies since the graphs are unlabeled, while for the others, we additionally interpret found anomalies using labels associated with the graphs and external sources. For example, we use the DBLP site to examine academic activities of authors.

5.7.2. Result. Following the strategy used in Akoglu et al. [2010], Kang et al. [2011], and Kang et al. [2014], we especially focus on relation between degrees and local triangle counts of nodes. There are two types of patterns of interest: a group of anomalous nodes with similar characteristics and a node far from a general pattern in the degree-triangle scatter plot.

OBSERVATION 2 (CORE-PERIPHERY IN WEB). *In* Stanford*, there are core-peripheries that can be divided into two subgroups: the first subgroup is a small dense graph and the other is a large sparse graph. The two subgroups are tightly connected.*

Figure 14a shows the result of discovering anomalous structures in the Stanford web graph. In the scatter plot of the degree and the local triangle count for the nodes, we observe several near clique structures shown as short vertical lines in the plot. They are not only tightly connected to each other but also share a large portion of neighbors outside the group. The number of those neighbors are relatively large and they are sparsely interconnected. As a result, each group and its neighbors form a core-periphery. Figure 14b shows the edge densities of the discovered groups and their neighbor groups, and the ratios of the group sizes over the neighbor sizes. The three groups that we discover show similar patterns.

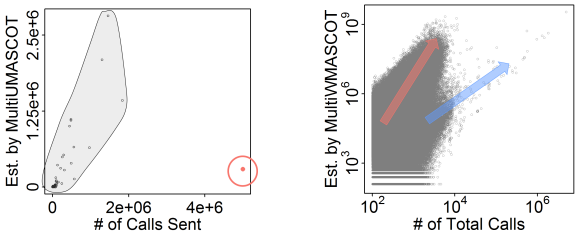
OBSERVATION 3 (BROAD POPULARITY OF MOVIES). *In* MovieRev9*, several sets of movies are loved by many users of various tastes.*

Figure 14c shows anomalous groups of nodes discovered in MovieRev9. The red group corresponds to religious films like “Holy Night” and “Return to Nazareth”; all of them are made with the same actors, writers and producers. Such movies can be preferred by various people believing in that religion regardless of their movie preferences, resulting in small triangles compared with degrees. The green, blue and purple groups correspond to classic movies, which are released in various forms and reissued until recently—the green for a series of “Planet of the Apes”, the blue for “Casablanca”, and the purple for “You Only Live Twice”. Since they have been loved a long period of time, regardless of preferences, many people with diverse spectra in their personalities watch those movies. As a result, those movies form less tightly connected ego networks.

OBSERVATION 4 (PAPERS BY MANY COAUTHORS). *In* DBLP-S*, there is a group of authors each of which participates in at least one paper with many coauthors, forming a small tightly connected group.*

OBSERVATION 5 (AMBIGUITY BY COMMON NAMES). *In* DBLP-S*, there are groups of very active researchers each of which corresponds to people having a same name.*

Figure 14d shows two linear patterns discovered in the coauthorship relations of DBLP-S. The first pattern is formed by the red and green groups which correspond to authors whose coauthors are tightly connected. We observe that this is due to papers written by a large number of coauthors. Especially, the green group contains 85 authors who participate in one paper written by 119 authors; some of them have only one publication.



(a) Telemarketers (b) Bimodality

Fig. 15. Anomaly detection results for PhoneCall. (a) There is a person (marked red) making calls to a huge number of people. That person can be construed as a telemarketer who calls arbitrary individuals. (b) There are two major patterns: "calling each other" (red) and "calling loosely connected people" (blue)

The second linear pattern marked in blue is observed on the area of large degrees and small local triangles. Those correspond to authors actively collaborating with other researchers, but we observe that each of them consists of a collection of a common name. Most of them are Chinese: e.g. the point with the largest degree is a result with many researchers whose names are "Wei Wang" expressed in 8 different words in Chinese [Sprouse 2007].

Below, we present results with MULTIBMASCOT and MULTIWMASCOT.

OBSERVATION 6 (TELEMARKETERS). *In* PhoneCall*, there is a person who makes a number of calls to many other people, but receives no call. Moreover, between the callees, there are few calls.*

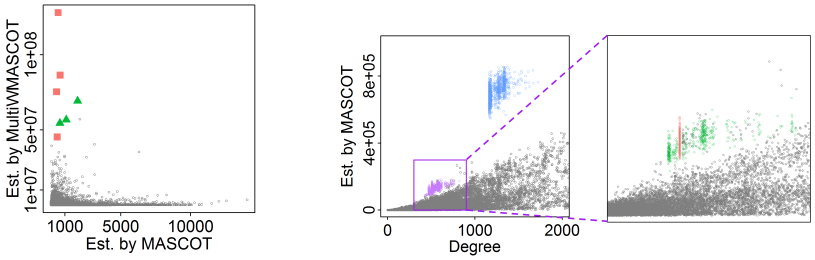
Figure 15a shows a person in PhoneCall who has numerous calls to many people between which a call is rare. It received no call while making over 5 millions calls. Such a pattern is likely to come from telemarketing, or from voice phishing in which a swindler contacts with a number of arbitrary people.

OBSERVATION 7 (BIMODALITY). *In* PhoneCall*, there are two major call patterns: calling each other and calling loosely connected people.*

Figure 15b shows two major patterns in PhoneCall. The first group in red corresponds to normal people who make calls with their friends, and most of the friends call each other. On the other hand, that pattern becomes weaker in the second group which is likely to be people in business: they make a number of calls with their business partners and customers who are unlikely to be friends with each other.

OBSERVATION 8 (INTENSIVE COLLABORATION). *In* DBLP-M*, there are groups of authors who wirte many papers with each other.*

Figure 16a shows two anomalous groups of nodes discovered in DBLP-M. We find that each group corresponds to co-authors writing many papers. For instance, we observe the red group whose four members appear with extremely high weighted local triangle counts on the y-axis, which is caused by intensive collaboration within themselves. Each of the four authors in the red group has three internal triangles that contribute 38%, 27%, 19%, and 50% to their total weighted triangle counts, but correspond to only 0.5%, 0.6%, 0.8%, and 0.7% of their total triangles, respectively. A similar pattern is also observed in the green group. A large portion of the weighted triangle counts of the



(a) DBLP-M (b) Petster

Fig. 16. Anomaly detection results for DBLP-M and Petster. (a) There are several groups in color consisting of co-authors for a lot of papers. The authors of each group intensively work with each other, resulting in many shared papers which lead to many weighted triangles. (b) There are two groups each of which is a (near) clique as well as well-separated from the others in Petster, marked in red and green.

three authors comes from the one internal triangle whose contribution is 50%, 51%, and 43% of the total, respectively.

OBSERVATION 9 (TIGHT COMMUNITIES). *In* Petster*, there are groups people who are tightly connected and well-separated from the others.*

Figure 16b shows two outlier groups of pet owners observed in Petster. The blue group consists of 1163 users forming a near clique, missing only six edges, who work at Dogster company. The employees are tightly connected to each other, leading to the group of relatively many triangles compared to their degrees. The purple points show another outliers in which two groups are mixed as shown in the right of Figure 16b. One major group is formed by the users in green who belong to the Dogster company as like the blue. The other group, marked as a bold vertical line in red, corresponds to 473 users forming a clique who raise a *Boxer*, a breed of a dog, in Hurst, Texas. The blue and red groups form a (near) clique well-separated from the rest of the graph. For a user in those groups, 94% and 93% of its neighbors belong to the same group while 6% and 7% of them belong to the rest of the graph on average, respectively.

6. CONCLUSION

In this paper, we propose local triangle counting algorithms for a graph stream: MASCOT for a simple graph, and MULTIWMASCOT and MULTIBMASCOT for a multigraph. The main contributions are as follows.

— **Algorithm.** We propose a one-pass local triangle counting algorithm MASCOT for a simple graph stream. MASCOT improves two basic algorithms MASCOT-C and MASCOT-A to provide both accuracy and memory-efficiency. We also propose MULTIBMASCOT and MULTIWMASCOT for a multigraph, which deal with unweighted and weighted triangle counting, respectively. MULTIBMASCOT is for the binary counting, and MULTIWMASCOT is for counting each triangle as the product of its three edge weights. All of the proposed algorithms require only one parameter of edge sampling probability and no prior knowledge on an input graph.

— **Performance.** Experimental results show that MASCOT is the best among our proposed algorithms for a simple graph stream and outperforms the existing one. Our evaluation also demonstrates that MULTIBMASCOT works as good as its corresponding simple graph algorithm though MULTIBMASCOT is directly applied to a multigraph without pre-processing, and MULTIWMASCOT works better for higher degree nodes.

— **Discovery.** Applying MASCOT to real graphs, we discover anomalous patterns, including the core-periphery structure in Web, a bimodal calling pattern in a phone call network, and intensive collaboration in DBLP.

Our future work includes developing streaming algorithms to solve various graph mining problems such as subgraph matching and graph partitioning.

ACKNOWLEDGMENTS

This work was supported by Research Resettlement Fund for the new faculty of Seoul National University. This work was also funded by Institute for Information communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No.R0190-15-2012, "High Performance Big Data Analytics Platform Performance Acceleration Technologies Development"). This work was also supported by AFOSR/AOARD under the Grant No. FA2386-16-1-4044. The ICT at Seoul National University provides research facilities for this study. The Institute of Engineering Research at Seoul National University provided research facilities for this work.

REFERENCES

Nesreen K. Ahmed, Nick G. Duffield, Jennifer Neville, and Ramana Rao Kompella. 2014. Graph sample and hold: a framework for big-graph analytics. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. 1446–1455.

Leman Akoglu, Mary McGlohon, and Christos Faloutsos. 2010. oddball: Spotting Anomalies in Weighted Graphs. In *Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part II*. 410–421.

Noga Alon, Raphael Yuster, and Uri Zwick. 1997. Finding and Counting Given Length Cycles. *Algorithmica* 17, 3 (1997), 209–223.

Shaikh Arifuzzaman, Maleq Khan, and Madhav V. Marathe. 2013. PATRIC: a parallel algorithm for counting triangles in massive networks. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*. 529–538.

Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. 2010. Fast Incremental and Personalized PageRank. *PVLDB* 4, 3 (2010), 173–184.

Bahman Bahmani, Ravi Kumar, Mohammad Mahdian, and Eli Upfal. 2012. PageRank on an evolving graph. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*. 24–32.

Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. 2002. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA*. 623–632.

Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. 2008. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*. 16–24.

Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. 2010. Efficient algorithms for large-scale local triangle counting. *TKDD* 4, 3 (2010).

Jonathan W Berry, Bruce Hendrickson, Randall A LaViolette, and Cynthia A Phillips. 2011. Tolerating the community detection resolution limit with edge weighting. *Physical Review E* 83, 5 (2011), 056119.

Luciana S. Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. 2006. Counting triangles in data streams. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*. 253–262.

Shumo Chu and James Cheng. 2011. Triangle listing in massive networks and its applications. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*. 672–680.

Jonathan Cohen. 2009. Graph Twiddling in a MapReduce World. *Computing in Science and Engineering* 11, 4 (2009), 29–41.

Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal. 2016. TRIÈST: Counting Local and Global Triangles in Fully-Dynamic Streams with Fixed Memory Size. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 825–834.

Prasanna Kumar Desikan, Nishith Pathak, Jaideep Srivastava, and Vipin Kumar. 2005. Incremental page rank computation on evolving graphs. In *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005 - Special interest tracks and posters*. 1094–1095.

Jean-Pierre Eckmann and Elisha Moses. 2002. Curvature of co-links uncovers hidden thematic layers in the world wide web. *Proceedings of the national academy of sciences* 99, 9 (2002), 5825–5829.

Xiaocheng Hu, Yufei Tao, and Chin-Wan Chung. 2013. Massive graph triangulation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*. 325–336.

Madhav Jha, C. Seshadhri, and Ali Pinar. 2013. A space efficient streaming algorithm for triangle counting using the birthday paradox. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*. 589–597.

Madhav Jha, C. Seshadhri, and Ali Pinar. 2015. A Space-Efficient Streaming Algorithm for Estimating Transitivity and Triangle Counts Using the Birthday Paradox. *TKDD* 9, 3 (2015), 15:1–15:21.

Hossein Jowhari and Mohammad Ghodsi. 2005. New Streaming Algorithms for Counting Triangles in Graphs. In *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, Kunming, China, August 16-29, 2005, Proceedings*. 710–716.

Daniel M. Kane, Kurt Mehlhorn, Thomas Sauerwald, and He Sun. 2012. Counting Arbitrary Subgraphs in Data Streams. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*. 598–609.

U. Kang, Brendan Meeder, and Christos Faloutsos. 2011. Spectral Analysis for Billion-Scale Graphs: Discoveries and Implementation. In *Advances in Knowledge Discovery and Data Mining - 15th Pacific-Asia Conference, PAKDD 2011, Shenzhen, China, May 24-27, 2011, Proceedings, Part II*. 13–25.

U. Kang, Brendan Meeder, Evangelos E. Papalexakis, and Christos Faloutsos. 2014. HEigen: Spectral Analysis for Billion-Scale Graphs. *IEEE Trans. Knowl. Data Eng.* 26, 2 (2014), 350–362.

Jinha Kim, Wook-Shin Han, Sangyeon Lee, Kyungyeol Park, and Hwanjo Yu. 2014. OPT: a new framework for overlapped and parallel triangulation in large-scale graphs. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*. 637–648.

Konstantin Kutzkov and Rasmus Pagh. 2013. On the streaming complexity of computing local clustering coefficients. In *Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4-8, 2013*. 677–686.

Matthieu Latapy. 2008. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theor. Comput. Sci.* 407, 1-3 (2008), 458–473.

Yongsub Lim and U. Kang. 2015. MASCOT: Memory-efficient and Accurate Sampling for Counting Local Triangles in Graph Streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. 685–694.

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.

Joel Nishimura and Johan Ugander. 2013. Restreaming graph partitioning: simple versatile algorithms for advanced balancing. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*. 1106–1114.

Rasmus Pagh and Francesco Silvestri. 2014. The input/output complexity of triangle enumeration. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*. 224–233.

Rasmus Pagh and Charalampos E. Tsourakakis. 2012. Colorful triangle counting and a MapReduce implementation. *Inf. Process. Lett.* 112, 7 (2012), 277–281.

Ha-Myung Park and Chin-Wan Chung. 2013. An efficient MapReduce algorithm for counting triangles in a very large graph. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*. 539–548.

Ha-Myung Park, Sung-Hyon Myaeng, and U. Kang. 2016. PTE: Enumerating Trillion Triangles On Distributed Systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 1115–1124.

Ha-Myung Park, Francesco Silvestri, U. Kang, and Rasmus Pagh. 2014. MapReduce Triangle Enumeration With Guarantees. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*. 1739–1748.

A. Pavan, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. 2013. Counting and Sampling Triangles from a Graph Stream. *PVLDB* 6, 14 (2013), 1870–1881.

Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy. 2011. Estimating PageRank on graph streams. *J. ACM* 58, 3 (2011), 13.

Thomas Schank and Dorothea Wagner. 2005. Finding, Counting and Listing All Triangles in Large Graphs, an Experimental Study. In *Experimental and Efficient Algorithms, 4th InternationalWorkshop, WEA 2005, Santorini Island, Greece, May 10-13, 2005, Proceedings*. 606–609.

Gene D Sprouse. 2007. Editorial: Which Wei Wang? *Physical Review Special Topics-Accelerators and Beams* 10, 12 (2007), 120001.

Isabelle Stanton. 2014. Streaming Balanced Graph Partitioning Algorithms for Random Graphs. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*. 1287–1301.

Isabelle Stanton and Gabriel Kliot. 2012. Streaming graph partitioning for large distributed graphs. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*. 1222–1230.

Siddharth Suri and Sergei Vassilvitskii. 2011. Counting triangles and the curse of the last reducer. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*. 607–614.

Charalampos E. Tsourakakis, Christos Gkantsidis, Bozidar Radunovic, and Milan Vojnovic. 2014. FENNEL: streaming graph partitioning for massive scale graphs. In *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*. 333–342.

Charalampos E. Tsourakakis, U. Kang, Gary L. Miller, and Christos Faloutsos. 2009. DOULION: counting triangles in massive graphs with a coin. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*. 837–846.

Howard T. Welser, Eric Gleave, Danyel Fisher, and Marc A. Smith. 2007. Visualizing the Signatures of Social Roles in Online Discussion Groups. *Journal of Social Structure* 8 (2007).

Zhi Yang, Christo Wilson, Xiao Wang, Tingting Gao, Ben Y. Zhao, and Yafei Dai. 2014. Uncovering social network Sybils in the wild. *TKDD* 8, 1 (2014), 2:1–2:29.

Received December 20xx; revised xxxx 20xx; accepted xxxx 20xx