MASCOT: Memory-efficient and Accurate Sampling for Counting Local Triangles in Graph Streams

Yongsub Lim
School of Computing
KAIST
yongsub@kaist.ac.kr

U Kang
School of Computing
KAIST
ukang@cs.kaist.ac.kr

ABSTRACT

How can we estimate local triangle counts accurately in a graph stream without storing the whole graph? The local triangle counting which counts triangles for 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 MASCOT, a memory-efficient and accurate method for local triangle estimation in a graph stream based on edge sampling. To develop MASCOT, we first present two naive local triangle counting algorithms in a graph stream: 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 an unconditional triangle counting for a new edge, regardless of whether it is sampled or not. In contrast to the existing algorithm which requires prior knowledge on the target graph and appropriately set parameters, MASCOT requires only one simple parameter, the 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. Thanks to MASCOT, we also discover interesting anomalous patterns in real graphs, like *core-peripheries* in the web and *ambiguous author names* in DBLP.

Categories and Subject Descriptors

G.2.2 [**Graph Theory**]: Graph algorithms; H.2.8 [**Database Applications**]: Data mining

General Terms

Design, Experimentation, Algorithms

Keywords

Local triangle counting; graph stream mining; edge sampling; anomaly detection

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.

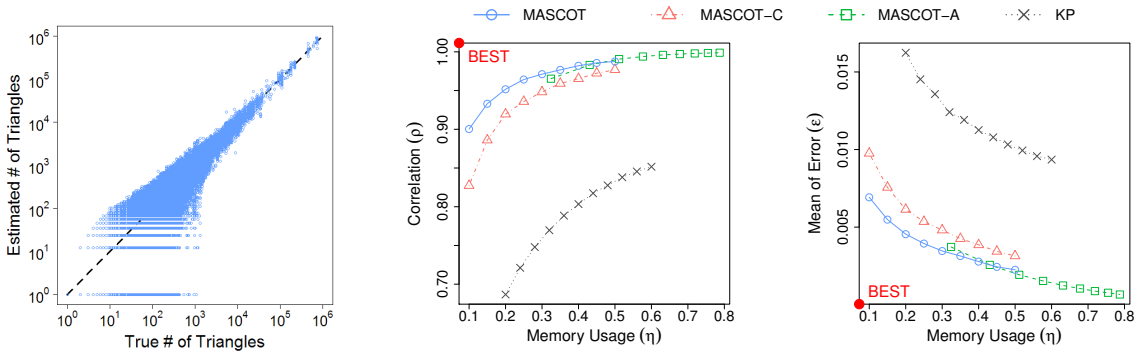
DOI: http://dx.doi.org/10.1145/2783258.2783285 .

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*? 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 [40]; the number of triangles of a user is examined to identify a social role of the user in the network [39], and is shown to be a good feature in assessing the content quality provided by the user [8]. In web mining, it is also used to find spam pages [8] and to uncover hidden thematic layers [15]. Other applications include network community detection [9] and motif detection in bioinformatics [24].

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 size graph. A number of algorithms to count triangles in a graph stream have been proposed [7, 8, 11, 17, 18, 19, 27]. 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 [22], 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.

In this paper, we propose MASCOT, a memory-efficient, and accurate one-pass local triangle counting algorithm for a graph stream based on edge sampling. MASCOT provides unbiased estimation of the number of local triangles for every node. We first develop two naive 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



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

Figure 1: Summary of our results on the BerkStan graph described in Table 3. (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 [22] 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.

Table 1: 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 accuracy.

	[Proposed]	Basic		Existing
	MASCOT	MASCOT-A	MASCOT-C	KP [22]
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

Table 2: 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
$\mathcal{N}_{uv}$	$N_u \cap N_v$
$\mathcal{T}$	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 $\triangle_u$
p	Default probability of sampling each edge
η	Ratio of sampled edge over the total edges
q_{uv}, q_e	True probability that an edge $e = (u, v)$ is sampled

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 1 compares MASCOT and other algorithms including KP [22] by various aspects.

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 [22] 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. Applying MASCOT 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 naive edge sampling based algorithms MASCOT-C and MASCOT-A to achieve both small variance and memory-efficiency. Unlike the previous algorithm [22], MASCOT requires only one simple parameter, the edge sampling probability, without requiring 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 [22] in terms of Pearson correlation coefficient and the mean of absolute relative error.
- **Discovery.** We discover several anomalous patterns in real graphs by applying MASCOT, including core-periphery structures in the web and ambiguous author names in a collaboration network.

The codes and data used in this paper are available at `http://kdmlab.org/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 naive ones in Section 3. After showing the performance of MASCOT in accuracy, and comparing it with the previous algorithm in Section 4, we conclude in Section 5.

2. RELATED WORK

In this section, we describe related works. Table 2 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 time-efficient algorithm is to use matrix multiplication [3]—A^3 for an adjacency matrix A results 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 [23, 32].

To handle large-scale graphs, researchers have been also interested in devising external and distributed algorithms. Chu and Cheng [12] proposed a graph partitioning based algorithm, and Hu et al. [16] developed an algorithm by iteratively loading a part of edges and counting triangles among them. Recently, Pagh and Silvestri [26] 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 [13] proposed the first MapReduce algorithm for triangle enumeration. Suri and Vassilvitskii [36] proposed a graph partitioning based algorithm, which is further improved by Park and Chung [28]. For more parallel algorithms, we refer to [4, 21, 29].

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. [7]. Afterwards, its space bound was improved by [18] and [11] successively. Recently, Jha et al. [17] 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. [19] 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 [30]. Tsourakakis et al. [38] 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 [1], 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. [8] proposed a semi-streaming algorithm requiring $\log n$ passes, and Kutzkov and Pagh proposed a single pass algorithm ("KP" henceforth) [22] based on node coloring [27]. However, they are limited in practice: [8] requires multiple passes to a graph, and [22] requires to know the number of nodes in a graph stream in advance. In contrast, our proposed Mascot requires only one pass to the graph with only one parameter, the edge sampling probability, while achieving a better accuracy. Note that although the one-pass local triangle counting methods, [22] and our proposed Mascot, assume no duplicated edges, both can be easily extended to a multigraph: e.g. duplicated edges can be checked using a hash scheme like Bloom filter [10].

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 [35] by suggesting several simple heuristics. Their method not only significantly outperforms a naive hashing scheme, but also is comparable to the state-of-the-art method for some cases. Stanton [34] theoretically showed the lower bound $o(n)$ of approximation ratio, and proposed two randomized greedy algorithms. We refer to [25, 37] for more related studies.

Another interesting topic is online PageRank computation. Sarma et al. [31] 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 [5, 6, 14].

3. PROPOSED METHOD

In this section, we propose Mascot, a memory-efficient accurate algorithm to provide unbiased estimation of local triangle counts for every node in a graph stream. We first present two naive algorithms Mascot-C and Mascot-A based on edge sampling. While Mascot-A provides lower variance than Mascot-C, it requires more memory spaces 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 [38], 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 $O(d_u + d_v)$ time where d_u is the degree of u. The whole procedure of Mascot-C is shown in Algorithm 1.

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

Lemma 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 with probability p, $\mathbb{E}\left[\delta_\lambda\right] = p^3$. Hence, we obtain $\mathbb{E}\left[\tau_u^c\right] = \triangle_u$. □

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

Input: Graph stream S, and edge sampling probability p.

$G \leftarrow (V, E)$ 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

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

Input: Graph stream S, and edge sampling probability p.

$G \leftarrow (V, E)$ with $V = E = \emptyset$.
foreach *edge* $e = (u, v)$ *from* S **do**
  ReadyNode(u).
  ReadyNode(v).
  if *e forms a triangle* **then**
    SampleEdgeA$(e, 1)$.
    CountTrianglesA(e).
  else
    SampleEdgeA(e, p).
  end
end
Function *SampleEdgeA*$((u, v), p)$
  $x \leftarrow Bernoulli(p)$.
  if $x = 1$ **then**
    $E \leftarrow E \cup \{(u, v)\}$.
    $q_{uv} = p$.
  end
end
Function *CountTrianglesA*$((u, v))$
  $N_{uv} \leftarrow N(u) \cap N(v)$.
  foreach $c \in N_{uv}$ **do**
    $s \leftarrow 1/q_{uv}q_{uc}q_{vc}$.
    $\tau_c \leftarrow \tau_c + s$.
    $\tau_u \leftarrow \tau_u + s$.
    $\tau_v \leftarrow \tau_v + s$.
  end
end

The following lemma analyzes variance of MASCOT-C.

LEMMA 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 $\mathcal{N}_{uv} = |\mathcal{N}_u \cap \mathcal{N}_v|$.

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

$$\mathrm{Var}\left[\tau_u^c\right] = \mathrm{Var}\left[\frac{1}{p^3}\sum_{\lambda \in \mathcal{T}_u} \delta_\lambda\right] = \frac{1}{p^6}\sum_{\lambda \in \mathcal{T}_u}\sum_{\gamma \in \mathcal{T}_u} \mathrm{Cov}\left[\delta_\lambda, \delta_\gamma\right]$$

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

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. □

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. 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 count of the nodes u, v and every $c \in \mathcal{N}_{uv}$. Since each edge has a different sampling probability, MASCOT-A maintains the sampling probability $q_{uv} \in \{p, 1\}$ for every sampled edge to give an appropriate weight for each sampled triangle. MASCOT-A is fully described in Algorithm 2. Like MASCOT-C, MASCOT-A provides unbiased estimation.

LEMMA 3. *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 $\mathbb{E}\left[\delta_\lambda\right] = q_e q_f q_g$. Consequently, we obtain $\mathbb{E}\left[\tau_u^a\right] = |\mathcal{T}_u| = \triangle_u$. □

For MASCOT-A, we analyze an upper bound of variance which is smaller than the variance of MASCOT-C in Lemma 2.

LEMMA 4. *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 $u \in V$,*

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

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

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

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

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

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

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

Since $\mathrm{Cov}\left[\delta_\lambda,\delta_\gamma\right] = 0$ 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:

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

- 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

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

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.

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. Based on this idea, we propose MASCOT as shown in Algorithm 3.

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

```
Input: Graph stream S, and
       edge sampling probability p.
G ← (V, E) with V = E = ∅.
foreach edge e = (u, v) from S do
    ReadyNode(u).
    ReadyNode(v).
    CountTriangles(e, 1/p^2).
    SampleEdge(e, p).
end
```

MASCOT provides unbiased estimation as follows.

LEMMA 5. *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 $u \in V$,*

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

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, $\mathbb{E}\left[\delta_\lambda\right] = p^2$. With the weight of $1/p^2$ for sampled edges, the lemma is proved by following the proof of Lemma 3. □

LEMMA 6. *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 $u \in V$,*

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

Table 3: 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
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 confliction
Gowalla[2]	196,591	950,327	Online social network
Stanford[2]	281,903	1,992,636	Web graph of Stanford.edu
NotreDame[2]	325,729	1,090,108	Web graph of Notre Dame
BerkStan[2]	685,230	6,649,470	Web graph of Berkeley and Stanford
LiveJournal[2]	4,846,609	42,851,237	LiveJournal online social network
MovieRev9[2]	253,045	6,611,899	Co-reviewed movies in Amazon
DBLP[1]	1,314,050	5,362,414	Co-author network in DBLP

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

PROOF. With $\mathbb{E}[\delta_\lambda] = p^2$, the proof is almost the same as that for Lemma 2. The only difference is to compute $\mathrm{Cov}[\delta_\lambda, \delta_\gamma]$. In MASCOT, $\mathbb{E}[\delta_\lambda \delta_\gamma]$ 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, $\mathbb{E}[\delta_\lambda \delta_\gamma] = p^4$; otherwise $\mathbb{E}[\delta_\lambda \delta_\gamma] = p^3$. Thus, $\mathbb{E}[\delta_\lambda \delta_\gamma] \leq p^3$, 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.

4. EXPERIMENTS

In this section, we show experimental results on performance of MASCOT and comparison with competing methods. Especially, we answer the following questions.

- Q1 How accurate is MASCOT?
- Q2 How better is MASCOT compared with the basic versions of MASCOT and the previous work [22]?
- Q3 How can MASCOT be applied to graph anomaly detection?

4.1 Dataset

We gather graph data from diverse domains such as social networks, router connectivity, hyperlinks in webpages, collaboration networks, citation networks, etc. We make them simple, i.e. directions, self-loops, and weights are removed. Their edges are given in a random order. Table 3 lists the datasets used in our experiments.

4.2 Evaluation Metric

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 $x_i^* = 0$.
- Ratio η of Sampled Edges: This dominates the amount of memory spaces required by an algorithm. It is equal to p for MASCOT and MASCOT-C, and larger than p for MASCOT-A in expectation.

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 [22], the same averaging scheme is used since the implementation is based on random hashing.

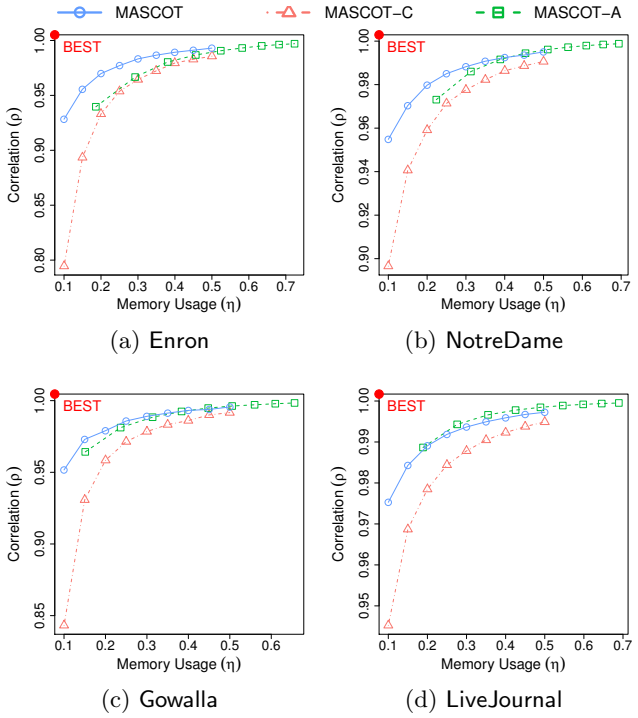
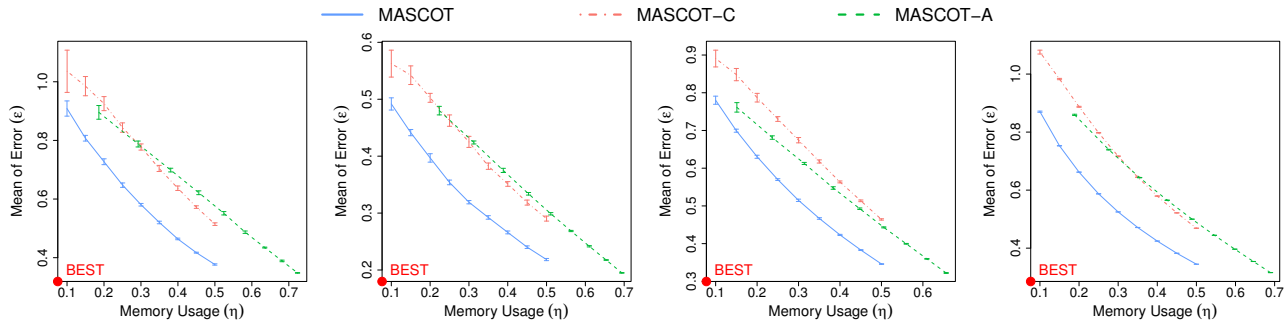


Figure 2: Pearson correlation coefficient ρ over different ratio η of sampled edges, which dominate the required memory spaces. MASCOT shows the best performance.

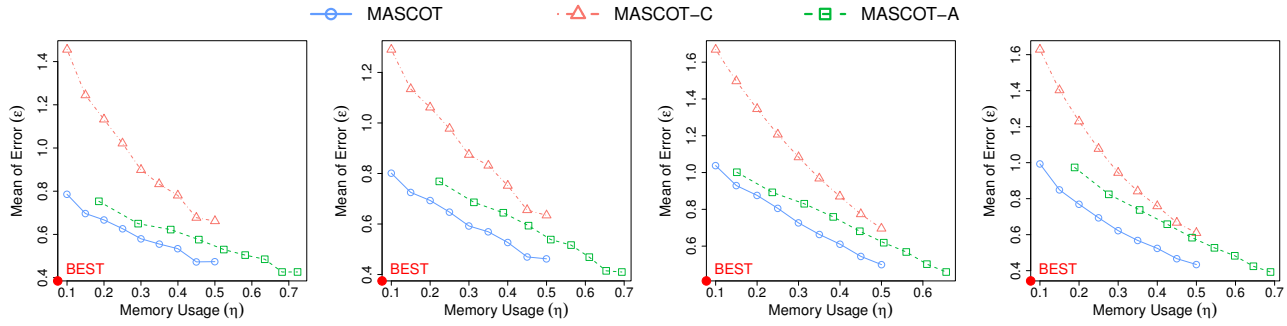
4.3 Performance of MASCOT

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



(a) Enron (b) NotreDame (c) Gowalla (d) LiveJournal

Figure 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.



(a) Enron (b) NotreDame (c) Gowalla (d) LiveJournal

Figure 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.

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.

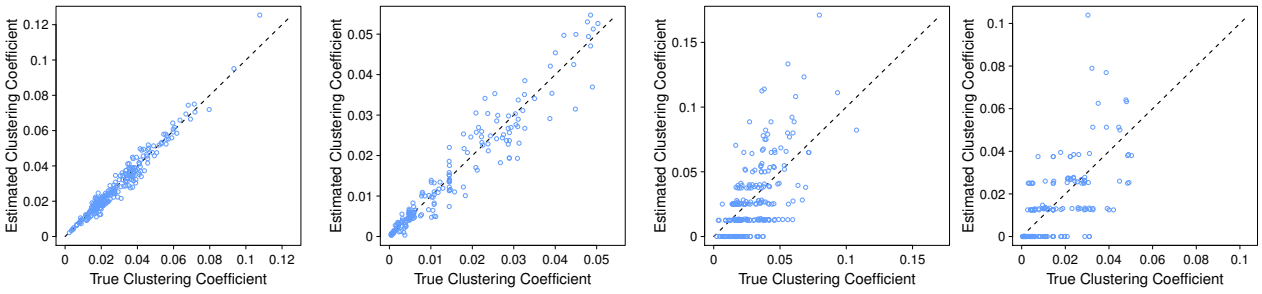
Figure 3 shows the mean ϵ of absolute relative error over the ratio η of sampled edges. Note that we also present the standard deviation of ϵ obtained by 10 runnings, as stated in Section 4.2, for every point. For all graphs, including those not shown in the figure, MASCOT works the best under the same 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.

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.

4.4 Comparison with Competing Method

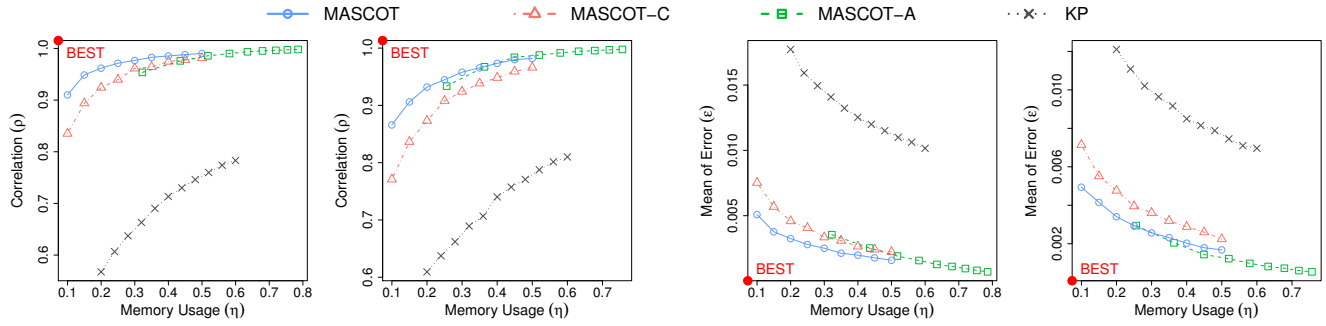
4.4.1 *Competing Method*

We consider the first and the only local triangle counting algorithm KP for a graph stream, proposed in [22]. KP generates multiple independent 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



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

Figure 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

Figure 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.

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 $\approx mK/C$.

4.4.2 *Comparison*

Setup. We compare MASCOT with KP for estimation 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[1]. Calculating the degree of each node in a graph stream is trivial, i.e. counting for each node its occurrences in the stream. For this experiment, we use the following three graphs: Stanford, Wiki-Conflict, and BerkStan.

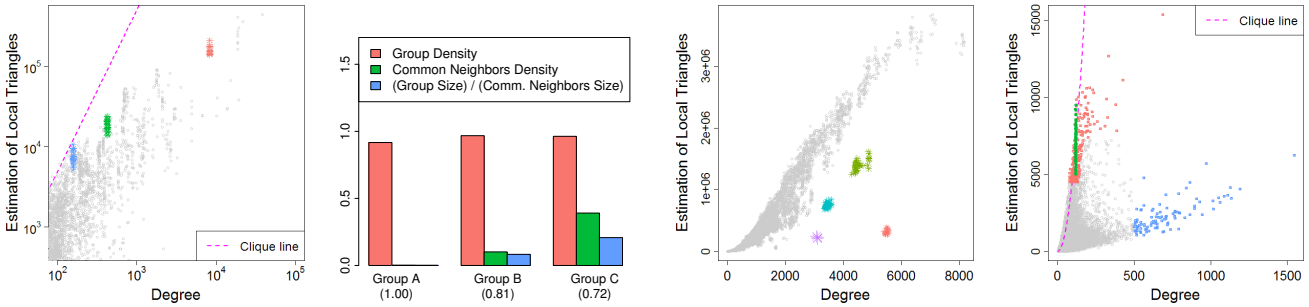
Result. Figures 1b, 1c and 5 show how well two methods MASCOT and KP estimate local clustering coefficients for the top-1000 high degree nodes. Note that all points for MASCOT are nearly on the $y = x$ line while KP's estimations are not that accurate despite somewhat positive correlations. Figure 6 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 mean and standard deviation.

Additionally, we compare MASCOT with the *semi-streaming* algorithm (BA) proposed in [8] for local triangle counting, which requires *multiple passes* to a graph. We set the number of passes to 50 and the number of random bits to $\log(n)$ for BA. We use the four graphs Advogato, Wiki-Conflict, BerkStan, and LiveJournal. In general, MASCOT is faster than BA, with larger absolute relative errors. However, for the top-1% high degree nodes, MASCOT is more accurate than BA in most cases. Furthermore, the correlation coefficient of MASCOT is better than BA in most cases.

4.5 Anomaly Detection in Graph Streams

It has been known that local triangle counts play an important role in determining characteristics of nodes [2, 8]. In this section, we show that our proposed algorithm MASCOT detects anomalous nodes or patterns in real world graph streams. We use the Stanford, MovieRev9 and DBLP graphs listed in Table 3.

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



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

Figure 7: Anomaly detection results for Stanford, Movie9 and DBLP. (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.

Setup. We use MASCOT with $p = 0.3$. Stanford is a hyperlink network of webpages: a node and an edge correspond to a web page and a hyperlink, respectively. In analysis of Stanford, we focus on structural anomaly since the graph is unlabeled. 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 if they participate in the same work. For all the graphs, we remove duplicated edges and edge directions.

Result. Following the strategy used in [2, 20], 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-triangles 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 7a 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 7b 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 7c 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*, 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*, there are groups of very active researchers each of which corresponds to people having a same name.*

Figure 7d shows two linear patterns discovered in the coauthorship relations of DBLP. 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.

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 [33].

5. CONCLUSION

In this paper, we propose MASCOT, a local triangle counting algorithm in a graph stream. The main contributions are as follows.

- **Algorithm.** We propose a one-pass local triangle counting algorithm MASCOT. MASCOT improves two basic algorithms MASCOT-C and MASCOT-A to provide both accuracy and memory-efficiency. In contrast to the previous algorithm [22], MASCOT requires 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 and outperforms the existing one.
- **Discovery.** Applying MASCOT to real graphs, we discover anomalous patterns like the core-periphery structure in Web, and ambiguity of author names in DBLP.

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

6. REFERENCES

[1] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella. Graph sample and hold: A framework for big-graph analytics. In *KDD*, 2014.

[2] L. Akoglu, M. McGlohon, and C. Faloutsos. oddball: Spotting anomalies in weighted graphs. In *PAKDD*, 2010.

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

[4] S. Arifuzzaman, M. Khan, and M. V. Marathe. Patric: a parallel algorithm for counting triangles in massive networks. In *CIKM*, 2013.

[5] B. Bahmani, A. Chowdhury, and A. Goel. Fast incremental and personalized pagerank. *PVLDB*, 4(3):173–184, 2010.

[6] B. Bahmani, R. Kumar, M. Mahdian, and E. Upfal. Pagerank on an evolving graph. In *KDD*, 2012.

[7] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *SODA*, 2002.

[8] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient algorithms for large-scale local triangle counting. *TKDD*, 4(3), 2010.

[9] J. W. Berry, B. Hendrickson, R. A. LaViolette, and C. A. Phillips. Tolerating the community detection resolution limit with edge weighting. *Phys. Rev. E*, 83:056119, 2011.

[10] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7), 1970.

[11] L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler. Counting triangles in data streams. In *PODS*, 2006.

[12] S. Chu and J. Cheng. Triangle listing in massive networks and its applications. In *KDD*, 2011.

[13] J. Cohen. Graph twiddling in a mapreduce world. *Computing in Science and Engineering*, 11(4):29–41, 2009.

[14] P. K. Desikan, N. Pathak, J. Srivastava, and V. Kumar. Incremental page rank computation on evolving graphs. In *WWW (Special interest tracks and posters)*, 2005.

[15] J.-P. Eckmann and E. Moses. Curvature of co-links uncovers hidden thematic layers in the World Wide Web. *PNAS*, 99(9):5825–5829, 2002.

[16] X. Hu, Y. Tao, and C.-W. Chung. Massive graph triangulation. In *SIGMOD*, 2013.

[17] M. Jha, C. Seshadhri, and A. Pinar. A space efficient streaming algorithm for triangle counting using the birthday paradox. In *KDD*, 2013.

[18] H. Jowhari and M. Ghodsi. New streaming algorithms for counting triangles in graphs. In *COCOON*, 2005.

[19] D. M. Kane, K. Mehlhorn, T. Sauerwald, and H. Sun. Counting arbitrary subgraphs in data streams. In *ICALP*, 2012.

[20] U. Kang, B. Meeder, E. Papalexakis, and C. Faloutsos. Heigen: Spectral analysis for billion-scale graphs. *TKDE*, 26(2):350–362, February 2014.

[21] J. Kim, W.-S. Han, S. Lee, K. Park, and H. Yu. Opt: A new framework for overlapped and parallel triangulation in large-scale graphs. In *SIGMOD*, 2014.

[22] K. Kutzkov and R. Pagh. On the streaming complexity of computing local clustering coefficients. In *WSDM*, 2013.

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

[24] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[25] J. Nishimura and J. Ugander. Restreaming graph partitioning: simple versatile algorithms for advanced balancing. In *KDD*, 2013.

[26] R. Pagh and F. Silvestri. The input/output complexity of triangle enumeration. In *PODS*, 2014.

[27] R. Pagh and C. E. Tsourakakis. Colorful triangle counting and a mapreduce implementation. *Inf. Process. Lett.*, 112(7):277–281, 2012.

[28] H.-M. Park and C.-W. Chung. An efficient mapreduce algorithm for counting triangles in a very large graph. In *CIKM*, 2013.

[29] H.-M. Park, F. Silvestri, U. Kang, and R. Pagh. Mapreduce triangle enumeration with guarantees. In *CIKM*, 2014.

[30] A. Pavan, K. Tangwongsan, S. Tirthapura, and K.-L. Wu. Counting and sampling triangles from a graph stream. *Proc. VLDB Endow.*, 6(14):1870–1881, Sept. 2013.

[31] A. D. Sarma, S. Gollapudi, and R. Panigrahy. Estimating pagerank on graph streams. *J. ACM*, 58(3):13, 2011.

[32] T. Schank and D. Wagner. Finding, counting and listing all triangles in large graphs, an experimental study. In *WEA*, 2005.

[33] G. D. Sprouse. Editorial: Which Wei Wang? *Physical Review Letters*, 99(23):230001, 2007.

[34] I. Stanton. Streaming balanced graph partitioning for random graphs. In *SODA*, 2014.

[35] I. Stanton and G. Kliot. Streaming graph partitioning for large distributed graphs. In *KDD*, 2012.

[36] S. Suri and S. Vassilvitskii. Counting triangles and the curse of the last reducer. In *WWW*, 2011.

[37] C. E. Tsourakakis, C. Gkantsidis, B. Radunovic, and M. Vojnovic. Fennel: Streaming graph partitioning for massive scale graphs. In *WSDM*, 2014.

[38] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos. Doulion: counting triangles in massive graphs with a coin. In *KDD*, 2009.

[39] H. T. Welser, E. Gleave, D. Fisher, and M. Smith. Visualizing the signatures of social roles in online discussion groups. *The Journal of Social Structure*, 8(2), 2007.

[40] Z. Yang, C. Wilson, X. Wang, T. Gao, B. Y. Zhao, and Y. Dai. Uncovering social network sybils in the wild. In *Internet Measurement Conference*, 2011.