Efficient and Adaptive Estimation of Local Triadic Coefficients

Ilie Sarpe
KTH Royal Institute of Technology
Stockholm, Sweden
ilsarpe@kth.se

Aristides Gionis
KTH Royal Institute of Technology
Stockholm, Sweden
argioni@kth.se

ABSTRACT

Characterizing graph properties is fundamental to the analysis and to our understanding of real-world networked systems. The *local clustering coefficient*, and the more-recent, *local closure coefficient*, capture powerful properties that are essential in a large number of applications, ranging from graph embeddings to graph partitioning. Such coefficients capture the local density of the neighborhood of each node, considering incident triangle structures and paths of size 2. For this reason, we refer to these coefficients collectively as *local triadic coefficients*.

In this work, we consider the novel and fundamental problem of efficiently computing the *average* of local triadic coefficients, over a given *partition* of the nodes of the input graph into a set of disjoint *buckets*. The *average local triadic coefficients* of the nodes in each bucket provide a better insight into the interplay of graph structure and the properties of the nodes associated to each bucket. Unfortunately, exact computation, which requires listing all triangles in a graph, is infeasible for large networks. Hence, we focus on obtaining *highly-accurate probabilistic estimates*.

We develop TRIAD, an adaptive algorithm based on sampling, which can be used to estimate the average local triadic coefficients for a partition of the nodes into buckets. TRIAD is based on a new class of unbiased estimators, and non-trivial bounds on its sample complexity, enabling the efficient computation of highly accurate estimates. Finally, we show how TRIAD can be efficiently used in practice on large networks, and we present a case study showing that average local triadic coefficients can capture high-order patterns over collaboration networks.

PVLDB Reference Format:
Ilie Sarpe and Aristides Gionis. Efficient and Adaptive Estimation of Local Triadic Coefficients. PVLDB, 18(8): 2561 - 2574, 2025.
doi:10.14778/3742728.3742748

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/iliesarpe/Triad.

1 INTRODUCTION

Graphs are a ubiquitous data abstraction used to study complex systems in different domains, such as social networks [17], protein interactions [8], information networks [56], and more [35]. A graph provides a simple representation: entities are represented by nodes and their relations are represented by edges, enabling the analysis of structural properties and giving unique insights into the function of various systems [63]. For example, flow analysis in transport graphs can be used for better urban design [20], subgraph patterns can improve recommenders [28]; and dense subgraph identification captures highly collaborative communities [31].

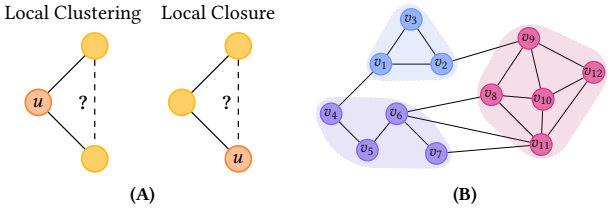

Figure 1: 1A: the local clustering coefficient of a node u considers the fraction of connected pairs of neighbors of u, while the local closure coefficient of u considers the fraction of neighbors of u's neighbors to which u is connected. 1B: a graph with its nodeset partitioned in three different sets.

The *local clustering coefficient* [60] is among the most important structural properties for graph analysis, and is used in many applications related to databases [13], social networks [18], graph embeddings [9], and link prediction [62]. The local clustering coefficient measures the fraction of connected pairs of neighbors of a given node u, e.g., see Figure 1A (left), providing a simple and interpretable value on how well the node is "embedded" within its local neighborhood. On an academic collaboration network, for example, the local clustering coefficient of an author u corresponds to the fraction of the coauthors of u collaborating with each other, capturing a salient coauthorship pattern of author u.

Recently, Yin et al. [65] introduced the *local closure coefficient*, a new coefficient capturing the fraction of paths of length two, originating from a node u that is closed by u, e.g., see Figure 1A (right). This novel definition directly accounts for the connections generated by u in the graph, differently from the local clustering coefficient that only depends on connections in u's neighborhood. The local closure coefficient is a simple concept that is gaining interest in the research community, as it provides additional and complementary insights to existing coefficients, and has applications in anomaly detection [68] and link prediction [65]. In this paper, we refer collectively to the local clustering coefficient and the local closure coefficient as *local triadic coefficients*. Both local triadic coefficients are fundamental quantities for graph analysis, as they capture structural properties of graphs on a local level [64–66].

In several applications, we are interested in the *average* local triadic coefficient of a subset of nodes [22, 32, 58]. The most typical

Proceedings of the VLDB Endowment, Vol. 18, No. 8 ISSN 2150-8097.
doi:10.14778/3742728.3742748

example is to consider the average over *all* nodes in the graph, e.g., the *average local clustering coefficient*: a standard graph statistic available in popular graph libraries [29, 43]. In general, computing the average local triadic coefficients of a *subset* of nodes based on graph properties can yield unique insights. For example, such properties may be associated with node metadata (e.g., in typed networks [52]), node similarities (e.g., from structural properties such core number and degrees, or role similarity [21]), or graph communities [23]. As a concrete example, in an academic collaboration network, we can compute the average local triadic coefficient of all authors who publish consistently in certain venues. For example, database conferences or machine learning—and average local coefficients could reveal interesting patterns for each community of interest, such as different trends in collaboration patterns.

The average local clustering coefficient is also exploited as a measure for community detection or clustering algorithms [36], with good quality clusters achieving high average local clustering coefficients, e.g., compared to random partitions. Hence, analyzing the average local triadic coefficient for different buckets can provide us with powerful insights for various applications, ranging from analyzing structural properties of specific groups of users with similar metadata, to community detection. Furthermore, average local triadic coefficients can also be employed to empower machine learning models, e.g., GNNs, for tasks such as graph classification or node embeddings [9].

Motivated by the previous settings requiring to compute the average local triadic coefficients over (given) sets of nodes, in this paper, we study the following problem: given a partition of the nodes of a graph into k sets, efficiently compute the average local triadic coefficient (clustering or closure) *for each* set of the partition.

Unfortunately, to address this problem, we cannot rely on exact algorithms, since exact computation of the local triadic coefficients for all graph nodes is an extremely challenging task and requires exhaustive enumeration of all triangles in a graph. Despite extensive study of exact algorithms for *triangle counting* [4, 5, 30], enumeration requires time $\Theta(m^{3/2})$, i.e., $\Theta(n^3)$ on dense graphs, which is extremely inefficient and resource-demanding for massive graphs.[1]

To overcome this challenge, we develop an efficient *adaptive approximation* algorithm: TRIAD (average local TRIadic ADaptive estimation), which can break the complexity barrier at the expense of a small approximation error. Similar to other approximation algorithms for graph analysis, TRIAD relies on *random sampling* [11, 59, 70]. Where, triangles incident to randomly sampled edges are used to update an estimate of the average triadic coefficient *for each* set of the partition of the nodes of a graph, through a novel *class* of unbiased estimators. TRIAD can approximate both the average local clustering and closure coefficients of arbitrary partitions of the graph nodes. Surprisingly, to the best of our knowledge, TRIAD is also the first algorithm specifically designed to estimate average local closure coefficients.

Our design of TRIAD is guided by two key properties, essential for many sampling schemes: (*i*) provide accurate estimates that are close to the unknown values being estimated; (*ii*) provide high-quality adaptive probabilistic guarantees on the distance between the estimates and the values being estimated. In fact, differently from existing approaches TRIAD quantifies the deviation between the probabilistic estimates reported in output and the underlying unknown average coefficients through a data-dependent approach [26, 27, 48, 69]. This results in an extremely efficient algorithm, which can adapt to the input graph since it is based on empirical quantities computed over the collected samples. Our contributions are as follows.

- We study the problem of efficiently obtaining high-quality estimates of the average local closure and average local clustering coefficients for each set in a partition of the nodes of a graph.
- We develop TRIAD, an efficient and adaptive algorithm providing high-quality estimates with controlled error probability. TRIAD is based on: a novel class of estimators that we optimize to achieve provably small variance—and a novel bound on the sample size obtained through the notion of pseudo-dimension. TRIAD also leverages state-of-the-art variance-aware concentration results to quantify the deviation of its estimates to the unknown estimated values, by the means of adaptive data-dependent bounds.
- We extensively assess TRIAD performances on large graphs showing that it provides high-quality probabilistic estimates and strong theoretical guarantees, not matched by existing state-of-the-art algorithms. We also show how the estimates of TRIAD can be used to study publication patterns among research fields over time on a DBLP graph.

2 PRELIMINARIES

Let $G = (V, E)$ be a simple and undirected graph with node-set $V = \{v_1, \ldots, v_n\}$ and edge-set $E = \{\{u, v\} : u \in V, v \in V \text{ and } u \neq v\}$, where $|V| = n$ and $|E| = m$.

For a node $v \in V$ we denote its *neighborhood* with $\mathcal{N}_v = \{u \in V :$ it exists $\{u, v\} \in E\}$ and its *degree* with $d_v = |\mathcal{N}_v|$. Similarly, given an edge $e = \{u, v\} \in E$ we define $\mathcal{N}_e = \{w \in V : w \in (\mathcal{N}_u \cap \mathcal{N}_v)\}$, i.e., the neighborhood of an edge $e \in E$ is the set of nodes from V that are neighbors to *both* incident nodes of e.

A *wedge* is a set of two distinct edges sharing a common node, i.e., $\mathrm{w} = \{e_1, e_2\} \subseteq E$ such that $|e_1 \cap e_2| = 1$; a wedge also corresponds to a path of length two. We let $\mathbb{W} = \{\mathrm{w} : \mathrm{w} \text{ is a wedge in } G\}$ be the set of all wedges in the graph G. Given a node $v \in V$ we say that a wedge $\mathrm{w} = \{e_1, e_2\}$ is *centered* at v if $\{v\} = e_1 \cap e_2$. The set of all such wedges is denoted with $\mathbb{W}_v^{\mathrm{c}}$. Note that, for each node $v \in V$, it holds $|\mathbb{W}_v^{\mathrm{c}}| = \binom{d_v}{2}$. A wedge $\mathrm{w} = \{e_1, e_2\}$ is *headed* at a node $v \in V$ if $v \in \mathrm{w}$ and $v \notin e_1 \cap e_2$.[2] The set of wedges headed at $v \in V$ is denoted with $\mathbb{W}_v^{\mathrm{h}}$. Note that $|\mathbb{W}_v^{\mathrm{h}}| = \sum_{u \in \mathcal{N}_v} (d_u - 1)$.

Example 2.1. Consider node v_8 from Figure 1B then $\{\{v_6, v_8\}, \{v_8, v_{11}\}\}$ is a wedge centered at v_8, and, $|\mathbb{W}_{v_8}^{\mathrm{c}}| = 6$ since $d_{v_8} = 4$. While $\{\{v_2, v_9\}, \{v_9, v_8\}\}$ is to a wedge headed at v_8 and $|\mathbb{W}_{v_8}^{\mathrm{h}}| = 13$.

Next, given a graph $G = (V, E)$ we define a *triangle* as a set of three edges that pairwise share an edge, i.e., $\delta = \{\{u, v\}, \{v, w\}, \{w, u\}\} \subseteq E$ and $u, v, w \in V$ are three distinct nodes. The set of all triangles in G is denoted by $\Delta = \{\delta : \delta \text{ is a triangle in } G\}$, while

[1] We use n and m for the number of nodes and edges respectively.

[2] We write $v \in \mathrm{w} = \{e_1, e_2\}$ to denote that $v \in e_1$ or $v \in e_2$.

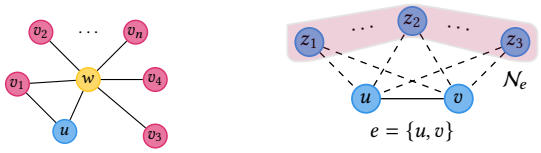

Figure 2: 2A: Discussion in Section 2. For node u it holds $\alpha_u = 1$ and $\phi_u = O(1/n)$. For node w it holds $\alpha_w = O(1/n^2)$ and $\phi_w = 1$. Thus, the local clustering and local closure coefficients can differ significantly. 2B: Discussion in Section 3. Consider a sampled edge $e \in E$. For each node $w \in V$ in the graph, our estimate for $|\Delta_w|$ is $|\Delta_w| = q|\Delta_e|/p$, if $w \in \{u, v\}$, and $|\Delta_w| = (1-2q)|\Delta_e|/p$, if $w = z_i \in \mathcal{N}_e$.

$\Delta_v = \{\delta \in \Delta : v \in \delta\} \subseteq \Delta$,[3] corresponds to the set of triangles containing $v \in V$. Similarly $\Delta_e = \{\delta \in \Delta : e \in \delta\}$ corresponds to the set of all triangles containing an edge $e \in E$.

We are now ready to introduce the fundamental triadic coefficients studied in this paper and introduced in earlier work [60, 65].

Definition 2.2. Given a graph $G = (V, E)$ and a node $v \in V$ we define the *local clustering coefficient* of v, denoted by α_v, and the *local closure coefficient* of v, denoted by ϕ_v, respectively as

$$\alpha_v = \frac{|\Delta_v|}{|\mathcal{W}_v^c|} = \frac{|\Delta_v|}{\binom{d_v}{2}} \quad \text{and} \quad \phi_v = \frac{2|\Delta_v|}{|\mathcal{W}_v^h|} = \frac{2|\Delta_v|}{\sum_{u \in \mathcal{N}_v}(d_v - 1)} \ .$$

As an example, consider v_9 in Figure 1B. Then $\alpha_{v_9} = 2/(2 \cdot 3) = 1/3$ and $\phi_{v_9} = (2 \cdot 2)/10 = 2/5$.

Observe that the values of α_v and ϕ_v for a node $v \in V$ can differ significantly, as illustrated in the example of Figure 2A.

Next, given a subset of nodes $V' \subseteq V$ we define the *average* local clustering coefficient (respectively, average local closure coefficient) as the average of the local clustering (respectively, local closure) coefficient of the nodes in the subset V', that is $\alpha(V') = \frac{1}{|V'|}\sum_{v \in V'} \alpha_v$ (respectively, $\phi(V') = \frac{1}{|V'|}\sum_{v \in V'} \phi_v$).

Given a set $A \neq \emptyset$, then $A_1, \ldots, A_k$ is a partition of A if $A_i \cap A_j = \emptyset$ for $i \neq j$ with $i, j \in [k]$, $A_i \neq \emptyset$ for all $i \geq 1$, and $\bigcup_{i \geq 1} A_i = A$. For ease of notation we denote a partition of V into k sets as $\mathcal{V}$. In Figure 1B we illustrate a network and a partition of the set of nodes. For ease of notation we use ψ_v to refer to a local triadic coefficient of a node $v \in V$ (that is, either clustering coefficient or closure coefficient) and $\Psi(V') = \frac{1}{|V'|}\sum_{v \in V'} \psi_v$.

Following standard ideas in the literature for controlling the quality of an approximate estimate [24, 41, 42], we consider two key properties: (*i*) the estimate should be close to the actual value; and (*ii*) there should exist rigorous guarantees quantifying the distance of the estimate to the unknown coefficient. These desirable properties are captured by the following problem formulation.

Problem 1. *Given a graph $G = (V, E)$, a partition $\mathcal{V}$ of the node-set V, a local triadic coefficient $\psi \in \{\alpha, \phi\}$, and parameters $(\varepsilon_j)_{j=1}^k \in (0, 1)$ and $\eta \in (0, 1)$, obtain:*

(A) *Estimates $f(V_j)$, for $j = 1, \ldots, k$, such that*

$$\mathbb{P}\left[\sup_{j=1,\ldots,k}\left|f(V_j) - \frac{1}{|V_j|}\sum_{v \in V_j}\psi_v\right| \geq \varepsilon_j\right] \leq \eta \ .$$

(B) Tight *confidence intervals C_j as possible, where $C_j = [f(V_j) - \widehat{\varepsilon}_j, f(V_j) + \widehat{\varepsilon}_j]$, for $j = 1, \ldots, k$, with $|C_j| = 2\widehat{\varepsilon}_j \leq 2\varepsilon_j$, such that over all k partitions it holds*

$$\mathbb{P}\left[\frac{1}{|V_j|}\sum_{v \in V_j}\psi_v \in C_j\right] \geq 1 - \eta \ .$$

To simplify our notation and when it is clear from the context we write f_j instead of $f(V_j)$ and Ψ_j instead of $\Psi(V_j)$, for $j \in [k]$.

In the remaining of this section we discuss the formulation of Problem 1. In particular, Problem 1 takes as input a graph G and a node partition $\mathcal{V}$. The goal is to obtain *accurate* estimates f_j, within at most ε_j *additive error* (i.e., $|f_j - \Psi_j| \leq \varepsilon_j$) for the local triadic coefficients Ψ_j with controlled error probability (η) *over all sets $j \in [k]$* of the partition $\mathcal{V}$. This requirement is enforced with condition (A) in Problem 1.

Furthermore, condition (B) ensures that the confidence intervals C_j (i.e., the ranges in which the values Ψ_j are likely to fall) centered around f_j are small. This requirement provides a rigorous guarantee on the proximity of f_j's to the corresponding Ψ_j's.

Note that Problem 1 allows the user to provide non-uniform error bounds ε_j, $j = 1, \ldots, k$. Such flexibility is highly desirable since (*i*) there may be partitions for which estimates are required with different levels of precision (as controlled by ε_j), e.g., in certain applications the user may require higher precision on the value of Ψ_j for some j's, and (*ii*) different values of ε_j may be required to distinguish between the values of Ψ_j for different sets in $\mathcal{V}$ [10, 39].

To clarify point (*ii*) above, consider $V = V_1 \cup V_2$ with $\Psi_1 = 10^{-2}$ and $\Psi_2 = 10^{-5}$. If $\varepsilon = \varepsilon_1 = \varepsilon_2 = 5 \cdot 10^{-2}$ then both $f_1, f_2 \in [0, \varepsilon]$ satisfy the guarantees of Problem 1 but this does not allow to distinguish between the very large value difference of Ψ_1 and Ψ_2 (i.e., three orders of magnitude). On the other hand, by allowing different accuracy levels, e.g., $\varepsilon_1 = 10^{-3}$ and $\varepsilon_2 = 10^{-4}$, we can address the issue. The acute reader may notice in this example we assume that we know the values of Ψ_1 and Ψ_2 in advance, and we set the values ε_j's accordingly, while in practice this is rarely the case. However, as we show in Section 3, our algorithm Triad can adaptively address the case of unknown Ψ_j's.

An alternative approach would be to require *relative error guarantees*, i.e., $|f_j - \Psi_j| \leq \varepsilon_j \Psi_j$, for all $j \in [k]$. Unfortunately this problem cannot be solved efficiently. First, it requires a lower bound on each value Ψ_j, and second it requires an impractical number of samples even for moderate values of Ψ_j and ε_j. In particular, state-of-the-art methods [14, 27, 48, 69] have shown that $\Omega(1/(\varepsilon\Psi_j)^2)$ samples may be needed. Thus, if, say, $\Psi = 10^{-3}$ and $\varepsilon = 10^{-2}$, then $\Omega(10^{10})$ samples would be needed, resulting in an extremely high and impractical running time.

3 METHODS

Our algorithm Triad consists of several components. At its core, it is based on a new class of unbiased estimators, which can be of independent interest for local triangle count estimation. We

[3] We write $v \in \delta$ to denote that there exists an edge $e \in \delta$ such that $v \in e$.

discuss such estimators in Section 3.1. We then introduce TRIAD, in Section 3.2. We present TRIAD's analysis in Section 3.3, and some practical optimizations in Section 3.4. We analyze the time and memory complexity of TRIAD in Section 3.5. Finally, in Section 3.6 we discuss the adaptive behavior of TRIAD. Missing proofs are reported in our extended version [46].

3.1 New estimates for local counts

We introduce a new *class* of estimators to approximate, for each node $v \in V$, the number of triangle counts $|\Delta_v|$ (i.e., locally to v), based on a simple sampling procedure that uniformly selects random edges. Such estimators stand at the core of the proposed method TRIAD, enabling a small variance of the estimates in output.

To compute the estimators for $|\Delta_v|$, for $v \in V$, first we sample uniformly an edge $e \in E$, and then collect the set of triangles Δ_e. The total number of triangles $|\Delta_e|$ is then used to estimate $|\Delta_v|$, *for each* node $v \in V$. The key, is in *how* the value $|\Delta_e|$ is distributed over all nodes $v \in e \cup \mathcal{N}_e$, to obtain an unbiased estimators of $|\Delta_v|$.

In particular, our estimators distribute the weight $|\Delta_e|$ *asymmetrically* across the nodes $v \in e$ (where $e \in E$ is the sampled edge) and the nodes $v \in \mathcal{N}_e$. This asymmetric assignment can be made before evaluating the estimates of $|\Delta_v|$, enabling us to obtain estimates with extremely small variance, as we show in Section 4.4.2, and discuss theoretically in Section 3.3.3.

Example 3.1. Consider Figure 2B, fix $q \in [0, 1/2]$, and suppose that $e = \{u, v\}$ is sampled. Then the estimators of $|\Delta_{z_i}|$ assign value $(1-2q)/p$ for all nodes $z_i \in \mathcal{N}_e$, and value $q|\Delta_e|/p$ for nodes $u, v \in e$, where p is the sampling probability of edge e.[4]

We will now show that the proposed estimates are unbiased with respect to $|\Delta_v|$, for all $v \in V$, and *any* value $q \in [0, 1/2]$,

LEMMA 3.2. *For any value of the parameter $q \in [0, 1/2]$,*

$$X_q(v) = \sum_{e \in E : v \in e} \frac{q|\Delta_e| X_e}{p} + (1-2q) \sum_{e \in E : v \in \mathcal{N}_e} \frac{X_e}{p} \qquad (1)$$

is an unbiased estimator of $|\Delta_v|$ for $v \in V$. That is, $\mathbb{E}[X_q(v)] = |\Delta_v|$, where the expectation is taken over a randomly sampled edge $e \in E$, and X_e is a 0–1 random variable indicating if $e \in E$ is selected.

Note that the estimator in Lemma 3.2 allows to flexibly select the parameter q, to minimize the variance of the estimates $X_q(v)$, for $v \in V$, leading to very accurate estimates with small variance when q is selected properly (see Section 3.3.3). We next use the result of Lemma 3.2 to obtain estimates $f_i(e) : E \mapsto \mathbb{R}_0^+$ of Ψ_i for each set V_i, with $i \in [k]$, of the partition $\mathcal{V}$ by sampling a random edge $e \in E$. First, to unify our notation, given a node $v \in V$ let $|\mathrm{W}_v^*| = |\mathrm{W}_v^{\mathsf{c}}|$ if $* = \mathsf{c}$, and $|\mathrm{W}_v^*| = |\mathrm{W}_v^{\mathsf{h}}|/2$ if $* = \mathsf{h}$. We can then write $\psi_v = |\Delta_v|/|\mathrm{W}_v^*|$, i.e., $\psi_v = \alpha_v$ corresponds to the local clustering coefficient if $* = \mathsf{c}$ and $\psi_v = \phi_v$ otherwise. Using this notation, we have the following.

LEMMA 3.3. *Let $e \in E$ be an edge sampled uniformly from E. Then, the random variable $f_j(e) : E \mapsto \mathbb{R}_0^+$ defined by*

$$f_j(e) = \sum_{e \in E} \frac{X_e}{p} \frac{1}{|V_j|} \sum_{v \in V_j} \frac{a_q(v,e)}{|\mathrm{W}_v^*|} \ ,$$

[4] In our analysis we consider $p = 1/m$, but any importance sampling probability distribution p_e over $e \in E$ can be used, provided that $p_e > 0$ if $|\Delta_e| > 0$.

Algorithm 1: TRIAD

Input: $G = (V, E)$, $(\varepsilon_j)_{j=1}^k$, η, partition $\mathcal{V}$, $\psi \in \{\alpha, \phi\}$.
Output: Estimates f_j and bounds $\widehat{\varepsilon}_j$ s.t. $|f_j - \Psi_j| \le \widehat{\varepsilon}_j \le \varepsilon_j$ for each $j \in [k]$ w.p. $> 1 - \eta$.

1 **if** $\psi = \alpha$ **then** $|\mathrm{W}_v^*| \leftarrow |\mathrm{W}_v^{\mathsf{c}}|$ for $v \in V$;
2 **else** $|\mathrm{W}_v^*| \leftarrow |\mathrm{W}_v^{\mathsf{h}}|/2$ for $v \in V$;
3 $f_j \leftarrow 0$ for each $j \in [k]$;
4 $q \leftarrow \texttt{Fixq}(G, \mathcal{V})$; $\varepsilon \leftarrow \min\{\varepsilon_j\}$;
5 $\zeta, R_1, \ldots, R_k \leftarrow \texttt{UpperBounds}(G, \mathcal{V}, |\mathrm{W}_v^*|_{v \in V}, q)$;
6 $i \leftarrow 0$; $\mathcal{S} \leftarrow \emptyset$; $s \leftarrow 0$; $\eta_0 \leftarrow \eta/2$; $R \leftarrow \max\{R_j, j \in [k]\}$;
7 $s_{\max} \leftarrow \frac{R^2}{\varepsilon^2}(\zeta + \log(1/\eta))$; $s_0 \leftarrow \lceil R \frac{3\log(4k/\eta_0)}{\varepsilon} + 1 \rceil$;
8 **while not** $\texttt{StoppingCondition}(s_{\max}, s, (f_j)_{j \ge 1}, \varepsilon)$ **do**
9 $\mathcal{S}_i \leftarrow \texttt{UniformSample}(E, s_i)$;
10 **foreach** $e = (u, v) \in \mathcal{S}_i$ **do**
11 **foreach** $w \in \mathcal{N}_e$ **do**
12 $f_j(e) \leftarrow f_j(e) + \frac{(1-2q)}{|\mathrm{W}_w^*|}$ such that $w \in V_j$;
13 $f_j(e) \leftarrow f_j(e) + \frac{q|\Delta_e|}{|\mathrm{W}_u^*|}$ such that $u \in V_j$;
14 $f_j(e) \leftarrow f_j(e) + \frac{q|\Delta_e|}{|\mathrm{W}_v^*|}$ such that $v \in V_j$;
15 $s \leftarrow s + s_i$; $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_i$;
16 $f_j \leftarrow \frac{m}{s|V_j|} \sum_{e \in \mathcal{S}} f_j(e)$ for each $j = 1 \ldots, k$;
17 $\widehat{\varepsilon}_j \leftarrow \texttt{ComputeEmpiricalBound}(\mathcal{F}, \mathcal{S}, \eta_i)$;
18 $i \leftarrow i + 1$; $\eta_i \leftarrow \eta_{i-1}/2$;
19 **return** $(f_j, \widehat{\varepsilon}_j)$ *for each* $j = 1 \ldots, k$;

where

$$a_q(v,e) = q|\Delta_e| \mathbf{1}[v \in e] + (1-2q) \mathbf{1}[v \in \mathcal{N}_e] \,, \qquad (2)$$

is an unbiased estimate of Ψ_j for each set $V_j \in \mathcal{V}$, $j \in [k]$, i.e., $\mathbb{E}[f_j(e)] = \Psi_j$. Here, X_e is a 0–1 random variable indicating if $e \in E$ is sampled, and $|\mathrm{W}_v^*| = |\mathrm{W}_v^{\mathsf{c}}|$ for $\psi = \alpha$, and $|\mathrm{W}_v^*| = |\mathrm{W}_v^{\mathsf{h}}|/2$ if $\psi = \phi$.

The proof can be found in the full version [46].

Lemma 3.3 shows that by sampling a single random edge $e \in E$ we can obtain an unbiased estimate of multiple coefficients Ψ_j associated sets $V_j \in \mathcal{V}$, by accurately weighting the triangles in Δ_e. This allows to *simultaneously* update the estimates of multiple buckets V_j, for $j \in [k]$, differently from existing approaches [24, 48, 69], where each triangle identified by the algorithm is used to estimate the coefficient of a *single* partition.

3.2 The TRIAD algorithm

Algorithm 1 presents TRIAD. The algorithm first initializes $|\mathrm{W}_v^*|$ for each $v \in V$ according to the coefficients ψ to be estimated (lines 1–2). TRIAD then proceeds to initialize the variables f_j, for $j = 1, \ldots, k$, corresponding to the estimates of Ψ_j to be output (line 3). It then sets ε to the smallest ε_j, for $j \in [k]$ (i.e., ε is the smallest upper bound ε_j required by the user, see Problem 1). TRIAD then selects the best value of the parameter $q \in [0, 1/2]$ to guarantee a fast termination of TRIAD by solving a specific optimization problem (line 4, subroutine `Fixq`, see Section 3.3.3). Intuitively, the value of q is fixed such that the maximum variance of f_j, $j \in [k]$ is minimized, yielding a fast convergence of TRIAD. In fact, as we will show, the

termination condition of TRIAD considers the *empirical* variance of the estimates f_j, over the sampled edges.

The function UpperBounds (line 5) computes R_j, for $j = 1, \ldots, k$, corresponding to bounds to the maximum value that a random variable $f_j(e)$ can take over a randomly sampled edge $e \in E$, i.e., $f_j(e) \leq R_j$ almost surely, for each $j \in [k]$. In addition UpperBounds computes ζ, an upper bound on the pseudo-dimension of the functions $f_j(e)$, $j \in [k]$. We use ζ and $R = \max_j R_j$ (line 6) to obtain a bound on the maximum number of samples $s_{\max}$ to be explored by TRIAD (line 7). We discuss the function UpperBounds, and prove the bound on the sample-size in Sections 3.3.4 and 3.3.1, respectively.

Auxiliary variables are then initialized (line 6): index i keeps track of the iterations, $\mathcal{S}$ maintains the bag of sampled edges processed, s counts the number of samples processed, and η_0 is used to obtain the probabilistic guarantees of TRIAD. Finally, s_0 corresponds to the initial sample size, i.e., the minimum sample size for which TRIAD can provide tight guarantees (line 7), that we discuss in Section 3.3.2.

The main loop of TRIAD is entered in line 8. At the i-th iteration of the loop, TRIAD samples a bag of s_i edges uniformly at random (line 9), and for each edge it computes $f_j(e)$ as defined in Lemma 3.3 (lines 12-14). Then TRIAD updates the sample size and the bag of edges processed (line 15) together with estimates f_j, for $j \in [k]$ (line 16) invoking the function ComputeEmpiricalBound to compute *non-uniform bounds* $\widehat{\varepsilon}_j$ on the deviations $|f_j - \Psi_j|$ such that $|f_j - \Psi_j| \leq \widehat{\varepsilon}_j$ with controlled error probability (line 17). These bounds are *empirical*, tight, and adaptive to the samples in $\mathcal{S}$, leveraging the variance of the estimates $f_j(e), e \in \mathcal{S}$, and the upper bounds R_j, $j \in [k]$ (see Sec. 3.3.2). Finally, a set of variables is computed for the next iteration, if the stopping condition of the main-loop is not met (line 18). The call to StoppingCondition in TRIAD returns "true" if one the following two conditions holds, which depend on the processed samples $\mathcal{S}$:
(1) if $s \geq s_{\max}$, then with probability at least $1 - \eta/2$ it holds: $|f_j - \Psi_j| \leq \varepsilon \leq \varepsilon_j$; (2) if $\widehat{\varepsilon}_j \leq \varepsilon_j$, then with probability at least $1 - \eta/2$ it holds: $|f_j - \Psi_j| \leq \widehat{\varepsilon}_j \leq \varepsilon_j$, for each $j \in [k]$.

When the main loop terminates TRIAD outputs (f_j, ε_j) in case (1), or $(f_j, \widehat{\varepsilon}_j)$ in case (2), for each set $V_j \in \mathcal{V}$. Note that our estimation addresses both requirements of Problem 1 as we guarantee that all the estimates f_j are within ε_j distance to Ψ_j, and furthermore, we report accurate error bounds $\widehat{\varepsilon}_j \leq \varepsilon_j$, for all $j \in [k]$, as $\Psi_j \in [f(V_j) - \widehat{\varepsilon}_j, f(V_j) + \widehat{\varepsilon}_j]$ with high probability. For example, even for as small ε_j as 10^{-2} the reported adaptive guarantees of TRIAD can be at most $\widehat{\varepsilon}_j \approx 10^{-3}$ in a few tens of seconds (see Section 4.3).

3.3 Analysis

In this section we present in more detail all the components of TRIAD, and we analyze its accuracy. The next lemma states that the estimates of Algorithm 1 are unbiased, which is important to prove tight concentration.

LEMMA 3.4. *For the output f_j, $j = 1, \ldots, k$, of Algorithm 1, it holds:*

$$\mathbb{E}[f_j] = \frac{1}{|V_j|} \sum_{v \in V_j} \psi_v = \Psi_j \ ,$$

that is, the estimates of Algorithm 1 are unbiased, for both local triadic coefficients $\psi \in \{\alpha, \phi\}$.

The proof can be found in the extended version [46].

Next we provide a bound for the variance of the estimates f_j, $j \in [k]$, returned by TRIAD.

LEMMA 3.5. *For the estimates f_j, $j = 1, \ldots, k$, of Algorithm 1, it is*

$$\mathrm{Var}[f_j] \leq \frac{1-p}{sp} \left(\frac{1}{|V_j|} \sum_{v \in V_j} \psi_v \right)^2 \ ,$$

for both local triadic coefficients $\psi \in \{\alpha, \phi\}$.

The proof can be found in the extended version [46].

3.3.1 Bounding the sample size. To present the bound on the sample size (i.e., $s_{\max}$), we first introduce the necessary notation. Given a finite domain $\mathcal{X}$ and $\mathcal{Q} \subseteq 2^{\mathcal{X}}$ a collection of subsets of $\mathcal{X}$,[5] a *range-space* is the pair $(\mathcal{X}, \mathcal{Q})$. We say that a set $X \subseteq \mathcal{X}$ is *shattered* by the range-set $\mathcal{Q}$, if it holds $\{Q \cap X : Q \in \mathcal{Q}\} = 2^X$. The *VC-dimension* $\mathrm{VC}(\mathcal{X}, \mathcal{Q})$ of the range-space is the *size* of the largest subset $X \subseteq \mathcal{X}$ such that X can be shattered by $\mathcal{Q}$. Given a family of functions $\mathcal{F}$ from a domain $\mathcal{H}$ with range $[a, b] \subseteq \mathbb{R}$, for a function $f \in \mathcal{F}$ we define the subset Q_f of the space $\mathcal{H} \times [a, b]$ as

$$Q_f = \{(x, t) : t \leq f(x)\}, f \in \mathcal{F} \ .$$

We then define $\mathcal{F}^+ = \{Q_f, f \in \mathcal{F}\}$ as the range-set over the set $\mathcal{H} \times [a, b]$. With these definitions at hand, the *pseudo-dimension* $\mathrm{PD}(\mathcal{F})$ of the family of functions $\mathcal{F}$ is defined as $\mathrm{PD}(\mathcal{F}) = \mathrm{VC}(\mathcal{H} \times [a, b], \mathcal{F}^+)$. For illustrative examples we refer the reader to the literature [41, 42, 50]. In our work, the domain $\mathcal{H}$ corresponds to the set of edges to be sampled by TRIAD to compute the estimators $f_j = f_{\mathcal{S},j}$, $j = 1, \ldots, k$, that is, $\mathcal{H} = E$.[6] We also define the set of functions $\mathcal{F} = \{f_{\mathcal{S},j} : j = 1, \ldots, k\}$, which corresponds to the set containing all functions f_j, $j = 1, \ldots, k$ in output to TRIAD.

THEOREM 3.6. *Let $\mathcal{F}$ be the set of functions defined above with $\mathrm{PD}(\mathcal{F}) \leq \zeta$, and let $\varepsilon, \eta \in (0, 1)$ be two parameters. If*

$$|\mathcal{S}| \geq \frac{(b-a)^2}{\varepsilon^2} \left(\zeta + \log \frac{1}{\eta} \right) \ ,$$

then with probability at least $1 - \eta$ over the randomness of the set $\mathcal{S}$ it holds that $\left| f_j - \Psi_j \right| \leq \varepsilon$, for each $j = 1, \ldots, k$.

Note that we cannot compute $\mathrm{PD}(\mathcal{F})$ from its definition as this requires exponential time in general, hence we now prove a tight and efficiently computable upper bound ζ such that $\mathrm{PD}(\mathcal{F}) \leq \zeta$. This enables us to use Theorem 3.6 and obtain a deterministic upper bound on the sample size of TRIAD (i.e., $s_{\max}$ in line 7).

First, let $\chi_v = |\{V_j \in \mathcal{V} : \text{exists } u \in (\mathcal{N}_v \cup \{v\}), u \in V_j\}|$, i.e., χ_v is the number of distinct sets V_j, for $j \in [k]$ from $\mathcal{V}$ containing a node in the set $\mathcal{N}_v \cup \{v\}$ for a given node $v \in V$.

Example 3.7. In the following extreme cases: (*i*) when every node is in a distinct bucket (i.e., $k = n$) then $\chi_v \leq d_{\max} + 1$, for $v \in V$, with $d_{\max}$ being the maximum degree of a node in V; (*ii*) when each node is in the same bucket then $\chi_v = 1$ for every node $v \in V$.

Now let

$$\widehat{\chi} = \max_{e=\{u,v\} \in E} \{\chi_z : z = \arg\min\{d_u, d_v\}\} \ . \tag{3}$$

[5] The set containing all possible subsets of $\mathcal{X}$.
[6] We write $f_{\mathcal{S},j}$ to explicit that a function depends on the bag of samples $\mathcal{S}$.

Intuitively $\widehat{\chi}$ corresponds to the largest number of buckets for which a sampled edge $e \in E$ yields a non-zero value for the estimate $f_j(e)$. Clearly, a trivial bound is $\widehat{\chi} \leq \min\{k, d_{\max}+1\}$, which can be very loose. We next present the bound on the pseudo-dimension associated to $\mathrm{PD}(\mathcal{F})$, recall that $\mathcal{F}$ corresponds to set of estimates $f_j, j = 1, \ldots, k$, of Triad.

Proposition 3.8. *The pseudo-dimension $\mathrm{PD}(\mathcal{F}) = \zeta$ is bounded as $\zeta \leq \lfloor \log_2 \widehat{\chi} \rfloor + 1$.*

The proof can be found in the extended version [46].

As an example of the powerful result of Theorem 3.8 consider the following corollary.

Corollary 3.9. *Fix $q = 0$ and take each node in a different set in $\mathcal{V}$ (i.e., $k = n$). Let G be a star graph. Then by Theorem 3.8 it holds $\zeta \leq 1$.*

The proof can be found in the extended version [46].

Corollary 3.9 shows that our result provides a tight bound on the pseudo-dimension associated to the family of functions $\mathcal{F}$. For comparison, under the setting of Corollary 3.9, then our setting maps to the one of de Lima et al. [14]. In their work, the authors prove $\zeta \leq O(\log n)$ for the graph in Corollary 3.9, while our result states $\zeta \leq 1$ yielding a $O(\log n)$ improvement, which is the maximum attainable. Clearly, a smaller upper bound for ζ implies a significantly smaller sample size required to guarantee the desired accuracy when such bound is used for Theorem 3.6.

We can refine the bounds on ζ for the values of $q = 0$ or $q = 1/2$. We discuss such refinement in our extended manuscript [46].

We conclude by noting that $\widehat{\chi}$ can be efficiently computed in $O(m)$ time complexity with a linear scan of the edges of the graph.

3.3.2 Computing adaptive error bounds. We detail how Triad leverages adaptive and variance-aware bounds to determine the distance of f_j from Ψ_j, for $j \in [k]$. We need a key concentration inequality.

Theorem 3.10 (Empirical Bernstein bound [33, 34]). *Let $X_1, \ldots, X_s$ be s independent random variables such that for all $i = 1, \ldots, s$, $\mathbb{E}[X_i] = \mu$ and $\mathbb{P}[X_i \in [a,b]] = 1$, and $\widehat{v} = \frac{1}{s}\sum_{i=1}^{s}(X_i - \overline{X}_s)^2$ where $\overline{X}_s = \frac{1}{t}\sum_{i=1}^{s} X_i$. Then,*

$$\left|\frac{1}{s}\sum_{i=1}^{s} X_i - \mu\right| \leq \sqrt{\frac{2\widehat{v}\log(4/\eta)}{s}} + \frac{7(b-a)\log(4/\eta)}{3(s-1)} \quad .$$

The above theorem connects the empirical variance over the samples $f_j(e)$ with the distance of f_j to Ψ_j, providing a powerful result. Therefore we can use Theorem 3.10 to obtain the function `ComputeEmpiricalBound`. That is, given a partition V_j and a bag of edges $\mathcal{S}$ sampled so far, we obtain $\widehat{\varepsilon}_j = \sqrt{\frac{2\widehat{v}_j \log(4k/\eta_i)}{s}} + \frac{7(R_j)\log(4k/\eta_i)}{3(s-1)}$ at iteration $i \geq 0$. Note that $\widehat{v}_j = \frac{1}{s}\sum_{e\in\mathcal{S}}(f_j(e) - f_j)^2$, which can be obtained in linear time (i.e., $|\mathcal{S}|$), assuming the values of $f_j(e)$ are retained over the iterations. Note that the above result provides also a criterion on how to set s_0. That is, s_0 should be at least $\lceil R\frac{3\log(4k/\eta_0)}{\varepsilon} + 1\rceil$ (see Triad in line 6) in the very optimistic case that the empirical variance $\widehat{v}_j = 0$, for each $j \in [k]$. We can now prove the guarantees offered by Triad.

Theorem 3.11. *The output of Triad $(f_j, \widehat{\varepsilon}_j)$, for $j \in [k]$, is such that with probability at least $1-\eta$ it is $|f_j - \Psi_j| \leq \widehat{\varepsilon}_j \leq \varepsilon_j$, simultaneously for all sets V_j, for $j \in [k]$.*

The proof can be found in the extended version [46].

3.3.3 Minimizing the variance. The value of the parameter q (in the estimator from Lemma 3.3) plays a key role for Triad, enabling extremely accurate estimates if set properly (see Section 4.4.2). There are many criteria to select the value of q, the most natural one would be to fix q such that Triad processes the minimal possible sample-size to terminate its main loop. Unfortunately, we cannot optimize directly for such property.

Instead, to select the value of q we minimizing the maximum variance of the functions f_j over all $j \in [k]$. In fact, a smaller variance of f_j allows Triad to terminate its loop by processing a small number of samples. To do so, we first sample a very small bag $\mathcal{S}$ of edges from E, and estimate the variance of the functions f_j over $\mathcal{S}$ *for each* possible value of q (see [46]). The variance of f_j can, in fact, be expressed as a quadratic equation in the parameter q, i.e., there exist efficiently computable values A_j, B_j, C_j such that $\widehat{V}_q(f_j) = A_j + B_j q + C_j q^2$. We detail in our extended version how such values are computed. Using such formulation, we can solve a quadratic optimization problem, minimizing the maximum variance over $\widehat{V}_q(f_j)$ over all possible values of $q \in [0, 1/2]$ and for each $j \in [k]$. We first show that a sufficiently small bag of samples can yield a good estimate of the variance $\widehat{V}_q(f_j)$.

Lemma 3.12. *There exist values A_j, B_j, and C_j obtained over $c \geq 2$ sampled edges such that $\mathbb{E}[\widehat{V}_q(f_j)] = \mathrm{Var}[f_j]$ for*

$$\widehat{V}_q(f_j) = \frac{m^2}{(c-1)|V_j|^2}\left(\sum_{e\in\mathcal{S}} A_j + qB_j + C_j q^2\right) \quad .$$

The proof can be found in the extended version [46].

The above lemma tells us that the estimates $\widehat{V}_q(f_j)$ provide an unbiased approximation of the variance $\mathrm{Var}[f_j]$ for each function f_j. The next lemma tells us that such estimates are also a good approximation, for each value of $q \in [0, 1/2]$.

Lemma 3.13. *There exist a small value $\varepsilon' > 0$ such that $\widehat{V}_q(f_j) \in [\mathrm{Var}[f_j] - \varepsilon', \mathrm{Var}[f_j] + \varepsilon']$ for each partition $j = 1, \ldots, k$ and value of $q \in [0, 1/2]$, with high probability.*

The proof can be found in the extended version [46].

To find the optimal value of q, given that we have a good estimate of $\mathrm{Var}[f_j]$, for each $j \in [k]$, we solve the following convex problem, minimizing the largest estimated variance $\widehat{V}_q(f_j)$ over $j \in [k]$.

Problem 2 (Optimization of q). *Let A_j, B_j and C_j be the coefficients contributing to estimate the variance $\widehat{V}_q(f_j)$ then solving the following quadratic program yields $x^* = q_{\mathtt{alg}}$ such that $\widehat{V}_{q_{\mathtt{alg}}}(f_j) \in \widehat{V}_{q^*}(f_j) \pm \varepsilon'$ for small ε', and controlled error probability, where q^* is the optimal value q minimizing the variance of functions $f_j, j = 1, \ldots, k$.*

$$\begin{aligned} &\textit{minimize} && t \\ &\textit{subject to} && t \geq A_j + B_j x + C_j x^2, \ \textit{for all} \ j = 1, \ldots, k, \\ &\textit{and} && x \in [0, 1/2] \ . \end{aligned}$$

Solving this quadratic problem with state-of-the-art solvers is efficient: x is in $\mathbb{R}$, the sample size c used to compute the terms A_j, B_j, C_j is a small constant, the objective function and the constraints are convex, and in practice k is small, yielding an overall efficient procedure (see Section 4.4). Note that in TRIAD we denote collectively with `Fixq` the procedure that computes both $\widehat{V}_q(f_j)$, for $j \in [k]$, and identifies $x^* = q_{\texttt{alg}}$.

3.3.4 `UpperBounds`. A detailed description of the `UpperBounds` subroutine, which requires linear time complexity in m, is reported our extended version [46].

The next lemma shows that `UpperBounds` outputs an upper bound on the ranges of $f_j(e), e \in E$, and ζ as from Proposition 3.8.

LEMMA 3.14. *The output of* `UpperBounds` *is such that $f_j(e) \leq R_j$ almost surely, for each $e \in E$, and partition V_j, $j = 1, \ldots, k$, where ζ corresponds to the pseudo-dimension bound from Proposition 3.8.*

3.4 Practical optimizations

In this section we introduce some practical optimizations to improve the performances of TRIAD, with minimal complexity overhead.

3.4.1 Improved empirical bounds. We can further refine the empirical bounds for terms $\widehat{\varepsilon}_j$, for $j \in [k]$ (in line 17) computed by TRIAD by leveraging the following result.

THEOREM 3.15 (PREDICTABLE PLUGIN-EMPIRICAL BERNSTEIN CONFIDENCE INTERVAL (PRPL-EB-CI) [51, 61]). *Let $X_1, \ldots, X_s$ be s i.i.d. random variables such that for all $i = 1, \ldots, s$, it is $\mathbb{E}[X_i] = \mu$ and $\mathbb{P}[X_i \in [a, b]] = 1$, with $R = |b - a|$, and let*

$$\omega(\lambda) = \frac{-\log(1 - R\lambda) - R\lambda}{4} \text{ for } \lambda \in [0, 1/R), \quad \widehat{\mu}_j = \frac{1}{j}\sum_{i=1}^{j} X_i,$$

$$\widehat{\sigma}_j = \frac{R^2/4 + \sum_{i=1}^{j}(X_i - \widehat{\mu}_{i-1})^2}{j}, \quad \lambda_{j,s} = \min\left\{\sqrt{\frac{2\log(2/\eta)}{s\widehat{\sigma}_{j-1}}}, \frac{1}{2R}\right\}$$

for $j = 1, \ldots, s$, where $\widehat{\mu}_0 = 0$ and $\widehat{\sigma}_0 = R^2/4$. Then it holds that

$$\left|\frac{\sum_{i=1}^{s}\lambda_{i,s}X_i}{\sum_{i=1}^{s}\lambda_{i,s}} - \mu\right| \leq \frac{\log(2/\eta) + (2/R)^2\sum_{i=1}^{s}[\omega(\lambda_{i,s})(X_i - \widehat{\mu}_{i-1})^2]}{\sum_{i=1}^{s}\lambda_{i,s}}$$

with probability at least $1 - \eta$.

While similar in spirit to Theorem 3.10, the above bound often yields a sharper empirical bound on the values $\widehat{\varepsilon}_j$. Note that the above estimator yields a slightly more complicated formulation, i.e., the output of TRIAD corresponds to $f_j = \frac{\sum_{i=1}^{s}\lambda_{i,s}f_j(e_i)}{\sum_{i=1}^{s}\lambda_{i,s}}$ where $f_j(e_i)$ is the estimate associated to bucket V_j, for $j \in [k]$ evaluated for the i-th sample from $\mathcal{S}$.

In addition, the stopping condition is evaluated using the bounds $\frac{\log(2/\eta)+(2/R)^2\sum_{i=1}^{s}[\omega(\lambda_{i,s})(X_i-\widehat{\mu}_{i-1})^2]}{\sum_{i=1}^{s}\lambda_{i,s}} = \widehat{\varepsilon}_j \leq \varepsilon_j$, for $j \in [k]$, and $f_j = \frac{1}{s}\sum f_j(e_i)$ if $s \geq s_{\max}$. In Section 4, we leverage Theorem 3.15, but for ease of notation and presentation we introduced TRIAD with the results of Theorem 3.10.

3.4.2 Fixed sample size variant. In this section we briefly describe a variant that we call TRIAD-F, which leverages a *fixed* sample size schema. That is, in many applications, concentration bounds (e.g., Theorem 3.10), even if tight and empirical, can still be conservative. Hence we modify TRIAD to leverage the novel estimators discussed in Lemma 3.4, and the adaptive procedure to select the value of q as described in Section 3.3.3, but we modify the stopping condition of TRIAD. That is, we only require TRIAD-F to process *at most* a number $s \geq 1$ samples as provided in input by the user. This is of interest in many applications where strictly sublinear time complexity is required. In fact, such modification strictly enforces a small number of samples to be processed. We show that such variant outputs highly accurate estimates *of each* bucket, by processing only 1‰ edges on most graphs (see Section 4.2).

3.4.3 Filtering very small degree-nodes. To enable better performance for TRIAD, we process the graph G by removing the nodes with *small* degree obtaining a graph G' where all the nodes' degrees, in G, are above a certain threshold. This step decreases the variance of the estimates computed by TRIAD, as small degree nodes can have high values for their metric ψ (i.e., close to 1), but sampling may perform poorly in approximating the values ψ, when computing the estimate f_j, for $j \in [k]$, similarly to what noted by de Lima et al. [14], Kutzkov and Pagh [26]. The key challenge is to bound the overall total work to $\Theta(n)$ time complexity, making such processing negligible, and retaining the correct information to recover the solution to Problem 1 on G.

Our approach identifies a threshold over the node degree distribution, namely β, for which computing all the nodes' triangles under such threshold requires at most linear time in n, i.e., bounded by Cn for a small fixed constant C. Obtaining Δ_v for a node $v \in V$ with degree d_v requires at most $O(d_v^2)$ time. Therefore we first identify the value β such that $\beta = \max_{i=1,\ldots,d_{\max}}$ such that $\sum_{j=1}^{i} j^2 D_j \leq Cn$, where D_i denotes the number of nodes in G such that their degree is exactly i, that is $D_i = |\{v \in V : d_v = i\}|$. Clearly, β can be computed in linear time $\Theta(n)$ by iterating all nodes, assuming constant access to their degrees d_v. Given the threshold β, for each node $v \in V$ with degree less than β we compute exactly the triangles Δ_v, keeping track for all the nodes $v \in V'$ that have degree higher than β of the triangles containing at least one removed node. We show that our approach is efficient and yields no additional estimation error for TRIAD in the extended version of the manuscript [46].

3.5 Time and memory complexity

Time complexity. We recall that the filtering step requires $O(n)$ total work, and the routine `UpperBounds` requires $O(km)$, while `Fixq` requires $O(d_{\max} + \mathsf{T}_{\mathrm{QP}})$. Note that T_{QP}, the complexity of solving the convex minimax problem, is negligible as the optimization is over k total convex non-integer constraints, and for our problem formulation we consider k as a (possibly large) constant.

Finally, the largest time complexity to perform the adaptive loop over $T = s_{\max}$ total iterations of TRIAD, can require up to $O(T(d_{\max} + T))$ time since: (1) each edge can be incident to $d_{\max}$ nodes; (2) the additional T^2 term is from computing the empirical variance at each iteration.[7] Hence the *worst* case complexity is $O(R^2\varepsilon^{-2}(\zeta + \log 1/\eta)(d_{\max} + T) + m)$. Note that such analysis is extremely pessimistic, in fact, in practice the complexity of TRIAD is instead close to $O(R\varepsilon^{-1}(\log k/\eta)d_{\max} + km)$, as we

[7] This complexity can be reduced to T by relying on the wimpy variance.

often observe TRIAD to terminate after a small number of iterations of its main loop, implying that the processed edges are at most $O(R\varepsilon^{-1}(\log k/\eta))$. Hence, when $O(R\varepsilon^{-1}(\log k/\eta))$ is a small fraction $\omega \ll 1$ of m (e.g., 1% of m) then the total complexity is bounded as $O(m(d_{\max}\omega + k))$, capturing the efficiency of TRIAD.

Memory complexity. The memory complexity of TRIAD is comparable to existing state-of-the-art methods [48] for estimating the local clustering coefficient, requiring $O(m)$ memory. In more detail, TRIAD requires memory $O(m+|\mathcal{S}|k+kn)$, where $|\mathcal{S}|$ is the size of the samples processed by TRIAD. Clearly $|\mathcal{S}| = s$ for TRIAD-F. When processing very large graphs with limited resources such complexity can be prohibitive, hence in such cases, we should rely on a (semi-)streaming or distributed (e.g., MPC) implementation of TRIAD [7, 24], an interesting future direction.

3.6 Adaptive guarantees

We briefly discuss the main advantages and limitations of TRIAD in the adaptive case. We observe that TRIAD has a significant advantage to solve Problem 1. Given a input graph G and a partition $\mathcal{V}$, TRIAD selects its estimators f_j, $j \in [k]$, by properly fixing the parameter q *adaptively*. That is, the parameter q is optimized by TRIAD directly on G, leading to estimators f_j, $j \in [k]$, with small variance as captured by our theoretical results in Lemma 3.13 and in practice in Section 4.2. TRIAD also adapts the number of processed samples ($|\mathcal{S}|$) by leveraging the results from Theorem 3.15. Such bounds involve the empirical variance $\widehat{\sigma}_j$, $j \in [k]$ optimized by TRIAD through the parameter q—and the upper bounds R_j on the range of $f_j(e)$, $j \in [k]$, $e \in E$. Where each R_j depends on the node distribution over $\mathcal{V}$ and G. Obtaining a non-trivial characterization of R_j is extremely challenging, but we observe the following. When the values R_j are large in practice the obtained empirical bounds $\widehat{\varepsilon}_j$ may be loose with respect to the actual deviations $|f_j - \Psi_j|$. Instead, when $R_j \in O(1)$ then TRIAD processes $\widetilde{O}(1/\varepsilon)$ samples (a significant improvement over $\widetilde{O}(1/\varepsilon^2)$).[8] This is an inherent trade-off, when $R \in O(1)$ then the runtime of TRIAD is almost constant providing tight bounds $\widehat{\varepsilon}_j \leq |f_i - \Psi_i|$, which depend on G and $\mathcal{V}$.

4 EXPERIMENTAL EVALUATION

In this section we present our extensive experimental evaluation. We start by first describing the setup, and then we discuss the results of our research questions.

4.1 Setting

Implementation details. We implemented our algorithms in C++20, and compiled it under gcc 9.4, with optimization flags. To solve the variance optimization problem (Section 3.3.3) we used Gurobi 11 under academic license. All the experiments were performed on a 72-core machine Intel Xeon Gold, running Ubuntu 20.04. The code to reproduce our results is publicly available.[9]

Datasets and partitions into buckets. For our experiments we consider multiple datasets available online, from medium to large sized, which are reported together with a summary of their statistics in Table 1. Details on the datasets including URLs are available in our extended version [46].

Table 1: Datasets used in the experimental evaluation. Statistics show: n the number of nodes, m the number of edges, $d_{\max}$ the maximum degree, $\bar{\alpha}$ (resp. $\bar{\phi}$) the average local clustering (resp. closure) coefficient over all nodes.

Dataset	n	m	$d_{\max}$	$\bar{\alpha}$	$\bar{\phi}$
fb-CMU	$6.6\,K$	$0.3\,M$	$8 \cdot 10^2$	0.27	0.12
SP	$1.6\,M$	$22\,M$	$1 \cdot 10^4$	0.11	0.03
FR	$12\,M$	$72\,M$	$3 \cdot 10^3$	0.08	0.01
OR	$3.1\,M$	$0.1\,B$	$3 \cdot 10^4$	0.17	0.06
LJ	$4.8\,M$	$43\,M$	$2 \cdot 10^4$	0.27	0.08
BM	$43\,K$	$14\,M$	$8 \cdot 10^3$	0.51	0.19
G500	$4.6\,M$	$0.1\,B$	$3 \cdot 10^5$	0.06	0.0
GP	$0.1\,M$	$12\,M$	$2 \cdot 10^4$	0.49	0.05
PT	$43\,K$	$44\,K$	$2 \cdot 10^1$	0.12	0.1
HW	$1.1\,M$	$56\,M$	$1 \cdot 10^4$	0.77	0.16
HG	$0.5\,M$	$13\,M$	$5 \cdot 10^4$	0.19	0.01
BNH	$0,7\,M$	$0.2\,B$	$2 \cdot 10^4$	0.5	0.3
TW	$0.2\,M$	$6.8\,M$	$4 \cdot 10^4$	0.16	0.01

For each dataset we considered three main different node partitions $\mathcal{V}$: (*i*) $\mathcal{V}_{core}$ is obtained by grouping nodes with similar core-number over a total of $k = 30$ buckets; (*ii*) $\mathcal{V}_{deg}$ is obtained by grouping together nodes with similar degrees over a total of $k = 25$ buckets; (*iii*) $\mathcal{V}_{logDeg}$ assigns each node to a bucket as function of its degree [24], i.e., a node with degree d is assigned to the bucket with index $\lfloor \log(1 + (d-2)/\log 2) \rfloor + 2$, hence it holds $k = O(\log n)$.

Note that all the above partition schemes place nodes with similar degree in the same bucket. This is often the case in practical applications, where nodes with similar degree are associated to similar structural functions [24].

In addition, To test a general input to Problem 1 we also consider two other partitions $\mathcal{V}$. Partition $\mathcal{V}_{rnd}$ assigns nodes at random into $k = 30$ buckets and $\mathcal{V}_{met}$ is obtained by clustering with $k = 10$ each graph using METIS [23]. We report the results obtained on these partitions in our extended version as they follow similar trends to the ones discussed below. We do not discuss the memory usage as it is similar to all algorithms (TRIAD uses slightly more space compared to baselines as from our analysis in Section 3.5).

Finally we use TRIAD-α (resp. TRIAD-ϕ) to denote TRIAD when used to approximate the average local clustering (resp. local closure) coefficient.

Research questions. Our experimental evaluation investigated the following research questions.

Q1. How TRIAD performs in terms of accuracy and efficiency when varying its sample size s? (Section 4.2)

Q2. How tight are the adaptive guarantees provided by TRIAD, compared to state-of-the-art approaches? (Section 4.3)

Q3. What is the runtime of TRIAD; what is the impact of parameter q, and what is the quality of our optimization of q? (Section 4.4)

Q4. Which patterns are captured by triadic coefficients and TRIAD over collaboration networks? (Section 4.5)

[8] In $\widetilde{O}(\cdot)$ we ignore logarithmic factors.
[9] https://github.com/iliesarpe/Triad.

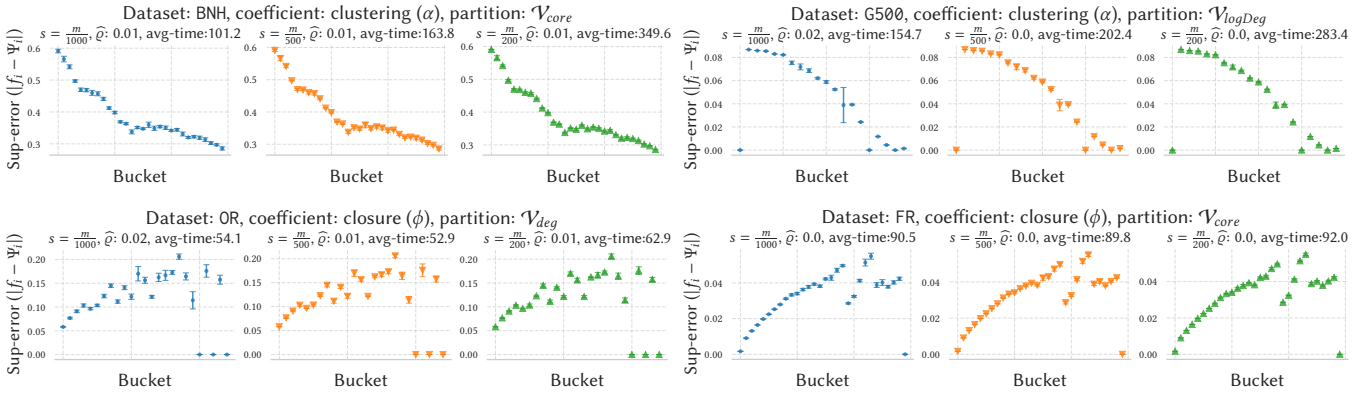

Figure 3: Value Ψ_i and its maximum error ($|f_i - \Psi_i|$) over five runs. We also report, the supremum error $\hat{\varrho} = \sup_{i \in [k]} |f_i - \Psi_i|$ and the average runtime over five independent runs across all buckets, for varying sample size ($s \in \{1, 2, 5\}$‰ of the total edges m).

4.2 Accuracy of estimates and efficiency

In this section we answer **Q1**, i.e., we study TRIAD's accuracy and efficiency by varying the sample size s. This setting is fundamental to show that TRIAD is both efficient and provides extremely accurate estimates by processing a small number of edges.

Setting. We consider TRIAD-F, which retains the adaptive selection of the parameter q while terminating TRIAD's main loop after processing exactly s samples (see Section 3.4.2). To set s we considered three different values: s = 1‰, s = 2‰, and s = 5‰ of the total edges m of each dataset. We then run each configuration (dataset, value of s, and bucket partition) for five independent runs. We then measure for each bucket the supremum error $|f_i - \Psi_i|$ over the five runs, and $\widehat{\varrho}$ the supremum of such errors across all buckets of $\mathcal{V}$. In addition, we measure the average runtime to process s samples over the five runs. Some representative results are presented in Figure 3.

Results. First, over almost all configurations tested we note that TRIAD's estimates are very accurate and tightly concentrated for each bucket of the various partitions. This is reflected by the supremum error $\widehat{\varrho}$, which is small and almost negligible even for very small sample sizes s such s=1‰. This holds in particular for datasets BNH and FR, while TRIAD requires a slightly higher sample size (i.e., s=2‰) to provide extremely accurate estimates for datasets G500 and OR. Note that the supremum error with a sample size of s=2‰·m tends to 0 on the considered configurations on all datasets. This supports the fact that TRIAD requires only a very small number of samples to obtain highly accurate estimates for Problem 1.

In addition, TRIAD's runtime is limited by at most a few hundred of seconds on very large datasets, yielding estimates almost comparable to the exact unknown values, showing that TRIAD is both efficient and highly accurate on both triadic coefficients. We report additional results under this setting in our extended version.

Summary. A very small sample size (of 1‰ total edges) is often sufficient to obtain highly accurate estimates for TRIAD, *simultaneously* over all buckets and different partitions for both the average local triadic coefficients, which is remarkable and extremely useful for highly-scalable network analysis.

4.3 Comparison with state-of-the-art

In this section we address **Q2**, i.e., we evaluate TRIAD and its adaptive guarantees with respect to existing state-of-the-art approaches.

Setting. We consider the state-of-the-art approach to approximate the local clustering coefficient values [48] (see Section 5), denoted with `WedgeSampler`-α (or WS-α for short). We extend the idea of wedge sampling to approximate the local closure coefficient, as there are no algorithms tailored for the local closure coefficient. This baseline, denoted with `WedgeSampler`-ϕ, is detailed in our extended version [46]. We fix $\varepsilon_j = \varepsilon = 0.075$ for all datasets and all buckets in $\mathcal{V}$ for TRIAD and $\eta = 0.01$, additional parameters are reported in our extended version. For each configuration we run TRIAD and obtain $\widehat{\varepsilon}_j$, i.e., the adaptive upper bounds on the distance between the estimates f_j and the unknown values Ψ_j for each bucket $V_j \in \mathcal{V}$. We then use such values as input for `WedgeSampler`, such that both algorithms provide the same guarantees. For each configuration we compute the estimation error as the supremum error $|f_j - \Psi_j|$, averaged over all buckets, our results will show the maximum of such supremum error over five runs. In addition, we report the average runtime for each algorithm on the various configurations, which was time-limited for all algorithms.

Results. Key results are summarized in Figure 4. We first observe that TRIAD reports very accurate estimates for Ψ_j on most configurations, which are much more precise than the ones provided by `WedgeSampler`. In particular, the supremum error, as desired, is of the order of 10^{-2} and on some configurations up to 10^{-3} (e.g., for datasets HG and OR). The baselines on most configurations achieve higher errors than TRIAD. We observe that this is especially the case for `WedgeSampler`-ϕ achieving higher error than TRIAD. Importantly, we observe that the range of improvement in accuracy over the baseline is up to one order of magnitude.

Remarkably, such results are obtained with a comparable or significantly smaller runtime with respect to the state-of-the-art baseline `WedgeSampler` (up to one order of magnitude on datasets HG, LJ and OR). In fact, our experiments confirm that TRIAD provides tighter bounds on the deviation between its estimates (f_j) and the unknown values (Ψ_j), significantly better than existing approaches,

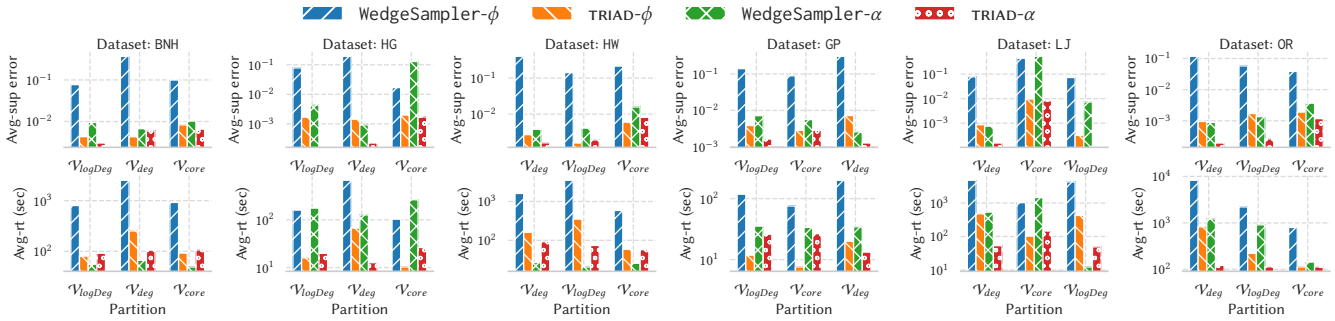

Figure 4: Comparison of TRIAD and the baselines WedgeSampler. For each dataset we show, (top plot): the average supremum error over all buckets over the various runs. (bottom): average runtime to perform an execution.

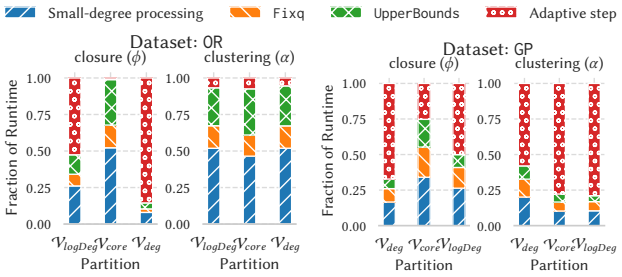

Figure 5: Fine grained runtime analysis. We show the average fraction of time spent in each step by TRIAD, the setting is from Section 4.3.

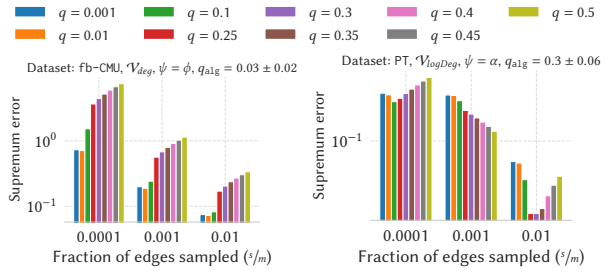

Figure 6: Supremum error over ten runs over different partitions, sample size s, coefficient ψ, and values of q.

while being more efficient. Unfortunately, in practice, such bounds may still be loose—this can noted by observing that we provided in input to TRIAD $\varepsilon_j = 0.075$ and under various datasets (e.g., HG, OR and LJ) the maximum errors for TRIAD are much smaller.

Summary. Our algorithm TRIAD provides better bounds on the deviation between the estimates f_j and the unknown coefficients Ψ_j, compared to existing approaches. While being tighter, such guarantees may still be loose for some settings, leaving an open question for future directions.

4.4 Runtime and parameter sensitivity

4.4.1 Runtime analysis. In this section we analyze TRIAD's runtime. In particular, we split the runtime into the following steps: (*i*) the practical optimization over the small-degree nodes; (*ii*) the optimization of the variance through Fixq; (*iii*) the execution of the routine UpperBounds; and (*iv*) the adaptive loop.

Figure 5 reports the average fraction of time spent by TRIAD in the various steps over the experiments from Section 4.3. We report two very different behaviors. On dataset OR, except for $\mathcal{V}_{\text{deg}}$, the time of performing the adaptive loop is negligible compared with all the other steps. This is in contrast with dataset GP, where most of TRIAD's runtime is spent in its adaptive loop, highlighting that TRIAD effectively adapts to the complexity of the graph in input. In other words, when the variance of the coefficients ψ_v is small, then the adaptive loop can terminate by processing a small amount of samples s as captured by Theorem 3.15.

Interestingly, we note that the procedure to optimize the variance (that we denote with Fixq) is often negligible, especially compared with UpperBounds, as captured by our analysis in Section 3.3.3.

Our results show that TRIAD's runtime depends on the complexity of the input graph i.e., the distribution of the unknown coefficients across buckets in $\mathcal{V}$, which allows TRIAD to compute highly accurate estimates very efficiently through its adaptive bounds.

4.4.2 Assessing the impact of q. In this section we investigate the impact of the parameter q on the quality of the estimates computed by TRIAD. Recall that q controls how the estimates f_j are computed, affecting the variance of the results. To evaluate such parameter, we selected two of our smallest datasets, for computational efficiency, and a fixed grid of nine values for the parameter q. For each value of q we then tested different sample sizes, i.e., using TRIAD-F with a sample size s such that $s/m \in \{0.0001, 0.001, 0.01\}$. For each combination of dataset, partition $\mathcal{V}$, value of q, and sample size s, we performed ten runs over both triadic coefficients $\psi \in \{\phi, \alpha\}$.

Some representative results are shown in Figure 6 reporting the supremum error (i.e., $\sup_j |f_j - \Psi_j|$) over the various configurations. We observe that the impact of q on the estimates f_j, over all sample sizes, can be from negligible (on the bottom-left plot) to very significant (bottom-right plot). In general, we observe a significant reduction in the supremum error by a proper selection

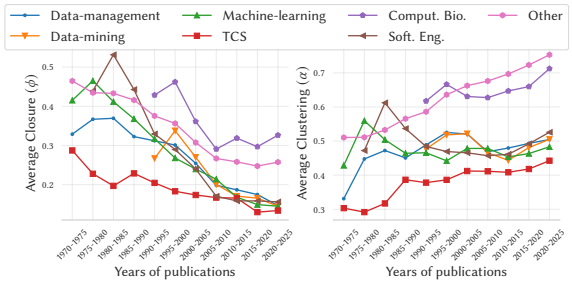

Figure 7: Average local clustering and local closure coefficient over the DBLP graph snapshots, for various computer science communities.

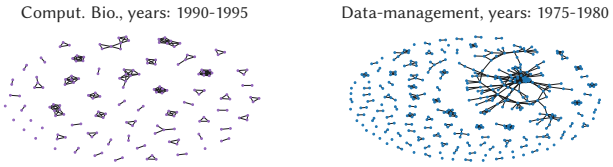

Figure 8: Induced subgraphs by communities. (Left): computational biology; (Right): data-management.

of the value of q—up to one order of magnitude in several settings (such as the top-left or bottom-right plots). This behavior confirms the importance of properly selecting the value of q. We note that the supremum error tends to be minimized with a specific value of q over the different settings, but this value is, in general, different on most configurations (e.g., top- and bottom-right plots).

Summarizing, a proper selection of the value of q can have a significant impact on the estimates of TRIAD leading to up to one order of accuracy in the estimates. The next section assesses how well our optimization aligns with a good value of the parameter q.

4.4.3 Optimization of q. We now briefly assess how well the value of q_{alg} (as optimized by TRIAD) aligns with a good choice for the parameter q. Results are shown in Figure 6, where we report the average q_{alg} and its standard deviation over ten runs. The size of the bag of samples used to compute q_{alg} is set to 500.

As captured by our analysis, the value of q_{alg} well-aligns with a good value of q obtained from the grid of tested values, on each configuration. In fact, q_{alg} often overlaps with the q yielding the minimum supremum error from the grid. For example, on the top-left plot the best value on the grid is $q = 0.01$ while $q_{\text{alg}} = 0.03 \pm 0.02$, highlighting that our method properly selects a good value for q, yielding small estimation variance.

4.5 Case study—academic collaborations

In this section we analyze collaboration patterns in different communities of the DBLP network using TRIAD, answering **Q4**.

Setting. DBLP collects bibliographic information about all major computer science journals and proceedings publications.[10] For each time-period of five consecutive years from 1970 to 2024, we collected the respective set of publications over DBLP. On each time period $t_1 = [1970, 1975], t_2 = [1975, 1980], \ldots$ we build a graph $G_{t_i} = (V_{t_i}, E_{t_i})$, with V_{t_i} consisting of authors, and edges corresponding to authors sharing a common publication. We then classified the authors, on each graph G_{t_i}, according to their research community. The resulting categories are reported in the legend of Figure 7, additional details on the classification are in in our extended version [46].

For each graph we computed the average triadic coefficients over each category. We then investigated if the analysis of the coefficients $f_j, j \in [k]$, provides us insights into similarities and differences of collaboration patterns over different communities.

Results. We observe in Figure 7 some interesting trends. For most of the communities, the average local clustering coefficient increases or remains stable over the years. Instead, the average local closure coefficient mostly decreases, for all but the computational-biology community. This can be explained by the fact that, new nodes over the network, are likely to have a small local closure coefficient when they collaborate with an author having many coauthors (e.g., students publishing with their advisor). Instead, it is easier for novel nodes on the network to a have higher local clustering coefficient, e.g., by collaborations within research groups. In addition, it is also easier for authors already belonging to the network to increase their local clustering coefficient over time, e.g., by publishing more.

There is a, perhaps surprising, increase in the average local closure coefficient for the computational-biology community over time. This could be explained by the fact that publications in this area require the joint effort of many authors, which likely increases the density of the connections of the authors in the graphs. Such an aspect can be observed in Figure 8, where we see that the structure of collaborations over the computational biology category forms many cliques. On the other hand, the structure of the graph of the data management community is more sparse, containing some chain structures. We further visualize the subgraph of other categories in our extended manuscript [46]. We conclude by noting that the average local clustering coefficient is significantly higher (ranging from 0.4 to more than 0.6) than the average local closure coefficient (which does not exceed 0.35) over all communities. This may be related to the nature of academic collaborations, where publishing with established researchers decreases the local closure coefficient of novel researchers on average. We further discuss that the above results cannot be uniquely explained by the degree distribution of the various nodes, in our extended version [46].

Summary. We analyzed the average local clustering (and closure) coefficients over different computer science communities across time. We observed that the values of the triadic coefficient can capture different collaboration patterns. For example, capturing highly collaborative (computational biology) and more sparse (data management) communities. Our findings show an example of a simple analysis using Problem 1 to gain better insights into publication and collaboration patterns in different research communities.

5 RELATED WORK

The problems addressed in this paper are closely related to counting triangles in graphs, which has been studied extensively. Thus, an

[10] https://dblp.org/

Table 2: Comparison of TRIAD and existing state-of-the-art approaches. Ψ: if the algorithm can estimate α, ϕ, or both. "Adaptive" denotes if the estimates adapt to the partitions $\mathcal{V}$. "Number of samples" denotes the number of samples processed. Finally "Processing complexity" denotes the time to process a sample, assuming $O(1)$ time complexity to check the existence of an edge. For ThinkD, r denotes the number of retained edges, which depends on ε^{-2}.

Algorithm	Ψ	Adaptive	Number of samples	Processing complexity
TRIAD	α, ϕ	✓	$\Omega\left(\frac{R\log k/\eta}{\varepsilon}\right)$ and $O\left(R^2\frac{(\zeta+\log 1/\eta)}{\varepsilon^2}\right)$	$O(d_{\max})$
WS-α [48]	α	✗	$\Theta\left(k\varepsilon^{-2}\log k/\eta\right)$	$O(1)$
WS-ϕ	ϕ	✗	$\Theta\left(k\varepsilon^{-2}\log k/\eta\right)$	$O(d_{\max})$
LCE [14]	α	✗	$\Theta\left(m^2\varepsilon^{-2}(\log d_{\max}+\log 1/\eta)\right)$	$O(d_{\max})$
ECC [26]	α	✗	$O(\varepsilon^{-2}\log n/\eta)$	$O(m\log\varepsilon^{-1})$
ThinkD [54]	α	✗	$O(r)$	$\Theta(m)$

extensive review is outside the scope [3, 49], instead, we focus only on discussing the most relevant problem settings and techniques.

Clustering and closure coefficient algorithms. The local clustering coefficient was first introduced by Watts and Strogatz [60]. Since their seminal work, many algorithms have been developed to efficiently compute related graph statistics. Many approaches consider both the approximation of the global clustering coefficient [12, 47], which is the local clustering coefficient averaged over all nodes in the graph, and the transitivity coefficient, which is the fraction of closed triangles over all the wedges in the graph [16], or weighted versions [27]. Interestingly, TRIAD can be adapted to compute all those coefficients with minimal modifications.

Many algorithms have been designed for computing the local clustering coefficient in restricted access models, such as (*i*) when the graph can be explored only through random walks [18]; (*ii*) when the graph is accessed in a (semi-)streaming fashion [7, 26]; or (*iii*) distributed environments [24]. Given that these works focus on restrictive scenarios, they require a large number of samples and often do not offer accurate guarantees.

The works most related to our formulation are by Etemadi and Lu [16], and Seshadhri et al. [48] who developed wedge-sampling algorithms, which, for each partition, sample wedges (i.e., our baseline considered in Section 4.3). These algorithms require a significantly high sample size, i.e., $\Theta(k\varepsilon^{-2}\log(k/\eta))$, which is tight [6]. Therefore these algorithms become impractical for small values of ε, as also demonstrated in our experiments.

Recently, de Lima et al. [14] developed an algorithm based on sampling edges and collecting their incident triangles, to approximate the local clustering coefficient of nodes with high-degree. The authors prove an upper bound on the sample complexity using VC-dimension. While their approach is similar in spirit to ours, it is significantly less general, i.e., their approach can be obtained by our class of estimators setting $q = 0$ in Equation (1). Their approach is also significantly less efficient, as their algorithm is based on data-independent bounds. In addition, our bound, as captured by Corollary 3.9, is significantly tighter than theirs.

Surprisingly, not much work has been done on algorithms for the closure coefficient. Recent works only quantify how this coefficient evolves in random networks [67].

Triangle-counting algorithms. As already noted, triangle counting is a wide area of research [4, 15, 25, 30, 57]. Most existing works address the problem of computing *global* triangle counts, and cannot therefore be used in our setting. Several works have been instead developed for *local* triangle counting. Exact methods [30, 37] are prohibitive for large networks, and sampling methods are designed for streaming settings [1, 53–55]. Those algorithms can provide accurate estimates for the local clustering coefficient when nodes have very high degree, while they are highly inaccurate for nodes with small degree. Note that this is a significant limitation for the setting we consider in this paper, where partitions may contain a large number of nodes with small degree.

Subgraph-counting algorithms. Another related problem is the one of counting subgraph occurrences, for which many different methods have been proposed [11, 19, 30, 40, 44, 45]. While these algorithms can be effectively used to count subgraph occurrences with respect to specific or multiple subgraph patterns, they cannot be easily adapted to extract high-quality local subgraph counts and average local triadic coefficients, as considered in Problem 1. Finally, Ahmed et al. [2] develop methods to estimate local subgraph counts, but compute *exactly* all triangles in the graph.

A summary of the key differences with most related works is reported in Table 2. Note that the last two algorithms are for streaming settings [26, 54], hence we did not consider them in Section 4, as they are designed for a more restrictive data-access model, yielding more inefficient methods. We observe from Table 2 that depending on the evaluation of the adaptive bounds on the given datasets TRIAD can be highly efficient (i.e., when R is small), improving significantly over existing methods as shown in Section 4.

6 CONCLUSION

We studied the problem of efficiently computing the average of local triadic coefficients. We designed TRIAD, an efficient and adaptive sampling algorithm. TRIAD estimates both the average local clustering coefficient and the recently-introduced average local closure coefficient, for which no algorithmic techniques were previously known. We showed that TRIAD is efficient and reports extremely accurate estimates, especially compared with existing methods.

There are several interesting directions for future work, such considering TRIAD for averages of local coefficients, which depend on the given partitions (e.g., a triangle is weighted differently if it contains nodes from different buckets), and weighted variants of the clustering and closure coefficients [27]. Another interesting direction is to adapt TRIAD for a multi-pass streaming setting [7, 26]. Finally, it will be interesting to study whether it is possible to design tighter bounds on the sample complexity, e.g., based on Rademacher complexity [38, 39].

ACKNOWLEDGMENTS

We thank Fabio Vandin for providing us the computing infrastructure. This research is supported by the ERC Advanced Grant REBOUND (834862), the EC H2020 RIA project SoBigData++ (871042), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

REFERENCES

[1] Nesreen K. Ahmed, Nick Duffield, Theodore L. Willke, and Ryan A. Rossi. 2017. On sampling from massive graph streams. *Proceedings of the VLDB Endowment* 10, 11 (Aug. 2017), 1430–1441. https://doi.org/10.14778/3137628.3137651

[2] Nesreen K. Ahmed, Theodore L. Willke, and Ryan A. Rossi. 2016. Estimation of local subgraph counts. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE. https://doi.org/10.1109/bigdata.2016.7840651

[3] Mohammad Al Hasan and Vachik S. Dave. 2017. Triangle counting in large networks: a review. *WIREs Data Mining and Knowledge Discovery* 8, 2 (Oct. 2017). https://doi.org/10.1002/widm.1226

[4] David A. Bader. 2023. Fast Triangle Counting. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 1–6. https://doi.org/10.1109/hpec58863.2023.10363539

[5] David A. Bader, Fuhuan Li, Zhihui Du, Palina Pauliuchenka, Oliver Alvarado Rodriguez, Anant Gupta, Sai Sri Vastav Minnal, Valmik Nahata, Anya Ganeshan, Ahmet Gundogdu, and Jason Lew. 2024. Cover Edge-Based Novel Triangle Counting. https://doi.org/10.48550/ARXIV.2403.02997

[6] Thomas Baignères, Pascal Junod, and Serge Vaudenay. 2004. *How Far Can We Go Beyond Linear Cryptanalysis?* Springer Berlin Heidelberg, 432–450. https://doi.org/10.1007/978-3-540-30539-2_31

[7] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. 2010. Efficient algorithms for large-scale local triangle counting. *ACM Transactions on Knowledge Discovery from Data* 4, 3 (oct 2010), 1–28. https://doi.org/10.1145/1839490.1839494

[8] Sourav S. Bhowmick and Boon Siew Seah. 2016. Clustering and Summarizing Protein-Protein Interaction Networks: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 28, 3 (March 2016), 638–658. https://doi.org/10.1109/tkde.2015.2492559

[9] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. 2019. Spectral Clustering with Graph Neural Networks for Graph Pooling. https://doi.org/10.48550/ARXIV.1907.00481

[10] Michele Borassi and Emanuele Natale. 2019. KADABRA is an ADaptive Algorithm for Betweenness via Random Approximation. *ACM Journal of Experimental Algorithmics* 24 (Feb. 2019), 1–35. https://doi.org/10.1145/3284359

[11] Marco Bressan, Stefano Leucci, and Alessandro Panconesi. 2019. Motivo: fast motif counting via succinct color coding and adaptive sampling. *Proceedings of the VLDB Endowment* 12, 11 (July 2019), 1651–1663. https://doi.org/10.14778/3342263.3342640

[12] Luciana S. Buriol, Gereon Frahling, Stefano Leonardi, and Christian Sohler. [n.d.]. *Estimating Clustering Indexes in Data Streams.* Springer Berlin Heidelberg, 618–632. https://doi.org/10.1007/978-3-540-75520-3_55

[13] Marek Ciglan, Alex Averbuch, and Ladialav Hluchy. 2012. Benchmarking Traversal Operations over Graph Databases. In *2012 IEEE 28th International Conference on Data Engineering Workshops*. IEEE, 186–189. https://doi.org/10.1109/icdew.2012.47

[14] Alane M. de Lima, Murilo V. G. da Silva, and André L. Vignatti. 2022. *Estimating the Clustering Coefficient Using Sample Complexity Analysis.* Springer International Publishing, 328–341. https://doi.org/10.1007/978-3-031-20624-5_20

[15] Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. 2017. Approximately Counting Triangles in Sublinear Time. *SIAM J. Comput.* 46, 5 (Jan. 2017), 1603–1646. https://doi.org/10.1137/15m1054389

[16] Roohollah Etemadi and Jianguo Lu. 2017. Bias correction in clustering coefficient estimation. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 606–615. https://doi.org/10.1109/bigdata.2017.8257976

[17] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2019. A survey of community search over big graphs. *The VLDB Journal* 29, 1 (July 2019), 353–392. https://doi.org/10.1007/s00778-019-00556-x

[18] Stephen J. Hardiman and Liran Katzir. 2013. Estimating clustering coefficients and size of social networks via random walk. In *Proceedings of the 22nd international conference on World Wide Web (WWW '13)*. ACM, 539–550. https://doi.org/10.1145/2488388.2488436

[19] Madhav Jha, C. Seshadhri, and Ali Pinar. 2015. Path Sampling. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. https://doi.org/10.1145/2736277.2741101

[20] Bin Jiang, Sijian Zhao, and Junjun Yin. 2008. Self-organized natural roads for predicting traffic flow: a sensitivity study. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 07 (July 2008), P07008. https://doi.org/10.1088/1742-5468/2008/07/p07008

[21] Ruoming Jin, Victor E. Lee, and Hui Hong. 2011. Axiomatic ranking of network role similarity. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '11)*. ACM, 922–930. https://doi.org/10.1145/2020408.2020561

[22] Marcus Kaiser. 2008. Mean clustering coefficients: the role of isolated nodes and leafs on clustering measures for small-world networks. *New Journal of Physics* 10, 8 (Aug. 2008), 083042. https://doi.org/10.1088/1367-2630/10/8/083042

[23] George Karypis and Vipin Kumar. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing* 20, 1 (Jan. 1998), 359–392. https://doi.org/10.1137/s1064827595287997

[24] Tamara G. Kolda, Ali Pinar, Todd Plantenga, C. Seshadhri, and Christine Task. 2014. Counting Triangles in Massive Graphs with MapReduce. *SIAM Journal on Scientific Computing* 36, 5 (jan 2014), S48–S77. https://doi.org/10.1137/13090729x

[25] Mihail N. Kolountzakis, Gary L. Miller, Richard Peng, and Charalampos E. Tsourakakis. 2012. Efficient Triangle Counting in Large Graphs via Degree-Based Vertex Partitioning. *Internet Mathematics* 8, 1-2 (mar 2012), 161–185. https://doi.org/10.1080/15427951.2012.625260

[26] Konstantin Kutzkov and Rasmus Pagh. 2013. On the streaming complexity of computing local clustering coefficients. In *Proceedings of the sixth ACM international conference on Web search and data mining (WSDM 2013)*, Vol. 5. ACM, 677–686. https://doi.org/10.1145/2433396.2433480

[27] Silvio Lattanzi and Stefano Leonardi. 2016. Efficient computation of the Weighted Clustering Coefficient. *Internet Mathematics* 12, 6 (June 2016), 381–401. https://doi.org/10.1080/15427951.2016.1198281

[28] Jure Leskovec, Ajit Singh, and Jon Kleinberg. 2006. *Patterns of Influence in a Recommendation Network.* Springer Berlin Heidelberg, 380–389. https://doi.org/10.1007/11731139_44

[29] Jure Leskovec and Rok Sosič. 2016. SNAP: A General-Purpose Network Analysis and Graph-Mining Library. *ACM Transactions on Intelligent Systems and Technology* 8, 1 (July 2016), 1–20. https://doi.org/10.1145/2898361

[30] Qiyan Li and Jeffrey Xu Yu. 2024. Fast Local Subgraph Counting. *Proceedings of the VLDB Endowment* 17, 8 (April 2024), 1967–1980. https://doi.org/10.14778/3659437.3659451

[31] Rong-Hua Li, Lu Qin, Jeffrey Xu Yu, and Rui Mao. 2015. Influential community search in large networks. *Proceedings of the VLDB Endowment* 8, 5 (Jan. 2015), 509–520. https://doi.org/10.14778/2735479.2735484

[32] Yusheng Li, Yilun Shang, and Yiting Yang. 2017. Clustering coefficients of large networks. *Information Sciences* 382–383 (March 2017), 350–358. https://doi.org/10.1016/j.ins.2016.12.027

[33] Andreas Maurer and Massimiliano Pontil. 2009. Empirical Bernstein Bounds and Sample Variance Penalization. (July 2009). https://doi.org/10.48550/ARXIV.0907.3740 arXiv:0907.3740 [stat.ML]

[34] Volodymyr Mnih, Csaba Szepesvári, and Jean-Yves Audibert. 2008. Empirical Bernstein stopping. In *Proceedings of the 25th international conference on Machine learning - ICML '08 (ICML '08)*. ACM Press, 672–679. https://doi.org/10.1145/1390156.1390241

[35] Mark Newman. 2018. *Networks.* Oxford University Press. https://doi.org/10.1093/oso/9780198805090.001.0001

[36] Xiaohui Pan, Guiqiong Xu, Bing Wang, and Tao Zhang. 2019. A Novel Community Detection Algorithm Based on Local Similarity of Clustering Coefficient in Social Networks. *IEEE Access* 7 (2019), 121586–121598. https://doi.org/10.1109/access.2019.2937580

[37] Noujan Pashanasangi and C. Seshadhri. 2020. Efficiently Counting Vertex Orbits of All 5-vertex Subgraphs, by EVOKE. In *Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM '20)*. ACM. https://doi.org/10.1145/3336191.3371773

[38] Leonardo Pellegrina. 2023. Efficient Centrality Maximization with Rademacher Averages. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM. https://doi.org/10.1145/3580305.3599325

[39] Leonardo Pellegrina and Fabio Vandin. 2023. SILVAN: Estimating Betweenness Centralities with Progressive Sampling and Non-uniform Rademacher Bounds. *ACM Transactions on Knowledge Discovery from Data* 18, 3 (Dec. 2023), 1–55. https://doi.org/10.1145/3628601

[40] Mahmudur Rahman, Mansurul Alam Bhuiyan, and Mohammad Al Hasan. 2014. Graft: An Efficient Graphlet Counting Method for Large Graph Analysis. *IEEE Transactions on Knowledge and Data Engineering* 26, 10 (Oct. 2014), 2466–2478. https://doi.org/10.1109/tkde.2013.2297929

[41] Matteo Riondato and Eli Upfal. 2018. ABRA: Approximating Betweenness Centrality in Static and Dynamic Graphs with Rademacher Averages. *ACM Transactions on Knowledge Discovery from Data* 12, 5 (July 2018), 1–38. https://doi.org/10.1145/3208351

[42] Matteo Riondato and Fabio Vandin. 2018. MiSoSouP: Mining Interesting Subgroups with Sampling and Pseudodimension. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. ACM. https://doi.org/10.1145/3219819.3219989

[43] Ryan Rossi and Nesreen Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. *Proceedings of the AAAI Conference on Artificial Intelligence* 29, 1 (March 2015). https://doi.org/10.1609/aaai.v29i1.9277

[44] Ryan A. Rossi, Nesreen K. Ahmed, Aldo Carranza, David Arbour, Anup Rao, Sungchul Kim, and Eunyee Koh. 2019. Heterogeneous Network Motifs. (Jan. 2019). https://doi.org/10.48550/ARXIV.1901.10026 arXiv:1901.10026 [cs.SI]

[45] Ryan A. Rossi, Anup Rao, Tung Mai, and Nesreen K. Ahmed. 2020. Fast and Accurate Estimation of Typed Graphlets. In *Companion Proceedings of the Web Conference 2020*. ACM. https://doi.org/10.1145/3366424.3382683

[46] Ilie Sarpe and Aristides Gionis. 2025. Efficient and Adaptive Estimation of Local Triadic Coefficients. *arXiv* (2025).

[47] Thomas Schank and Dorothea Wagner. 2005. Approximating Clustering Coefficient and Transitivity. *Journal of Graph Algorithms and Applications* 9, 2 (2005), 265–275. https://doi.org/10.7155/jgaa.00108

[48] C. Seshadhri, Ali Pinar, and Tamara G. Kolda. 2014. Wedge sampling for computing clustering coefficients and triangle counts on large graphs. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 7, 4 (may 2014), 294–307. https://doi.org/10.1002/sam.11224

[49] Comandur Seshadhri and Srikanta Tirthapura. 2019. Scalable Subgraph Counting: The Methods Behind The Madness. In *Companion Proceedings of The 2019 World Wide Web Conference (WWW '19)*. ACM, 1317–1318. https://doi.org/10.1145/3308560.3320092

[50] Shai Shalev-Shwartz. 2014. *Understanding machine learning*. Cambrige University Press, Cambridge. Hier auch später erschienene, unveränderte Nachdrucke.

[51] Shubhanshu Shekhar and Aaditya Ramdas. 2023. On the near-optimality of betting confidence sets for bounded means. https://doi.org/10.48550/ARXIV.2310.01547

[52] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. 2017. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (Jan. 2017), 17–37. https://doi.org/10.1109/tkde.2016.2598561

[53] Kijung Shin. 2017. WRS: Waiting Room Sampling for Accurate Triangle Counting in Real Graph Streams. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1087–1092. https://doi.org/10.1109/icdm.2017.143

[54] Kijung Shin, Sejoon Oh, Jisu Kim, Bryan Hooi, and Christos Faloutsos. 2020. Fast, Accurate and Provable Triangle Counting in Fully Dynamic Graph Streams. *ACM Transactions on Knowledge Discovery from Data* 14, 2 (Feb. 2020), 1–39. https://doi.org/10.1145/3375392

[55] Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal. 2017. TRIÈST: Counting Local and Global Triangles in Fully Dynamic Streams with Fixed Memory Size. *ACM Transactions on Knowledge Discovery from Data* 11, 4 (June 2017), 1–50. https://doi.org/10.1145/3059194

[56] Yizhou Sun and Jiawei Han. 2013. Mining heterogeneous information networks: a structural analysis approach. *ACM SIGKDD explorations newsletter* 14, 2 (2013), 20–28.

[57] Charalampos E. Tsourakakis, U. Kang, Gary L. Miller, and Christos Faloutsos. 2009. DOULION. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. https://doi.org/10.1145/1557019.1557111

[58] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. 2011. The Anatomy of the Facebook Social Graph. https://doi.org/10.48550/ARXIV.1111.4503

[59] Pinghui Wang, Junzhou Zhao, Xiangliang Zhang, Zhenguo Li, Jiefeng Cheng, John C.S. Lui, Don Towsley, Jing Tao, and Xiaohong Guan. 2018. MOSS-5: A Fast Method of Approximating Counts of 5-Node Graphlets in Large Graphs. *IEEE Transactions on Knowledge and Data Engineering* 30, 1 (Jan. 2018), 73–86. https://doi.org/10.1109/tkde.2017.2756836

[60] Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 393, 6684 (jun 1998), 440–442. https://doi.org/10.1038/30918

[61] Ian Waudby-Smith and Aaditya Ramdas. 2023. Estimating means of bounded random variables by betting. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 86, 1 (Feb. 2023), 1–27. https://doi.org/10.1093/jrsssb/qkad009

[62] Zhihao Wu, Youfang Lin, Jing Wang, and Steve Gregory. 2016. Link prediction with node clustering coefficient. *Physica A: Statistical Mechanics and its Applications* 452 (June 2016), 1–8. https://doi.org/10.1016/j.physa.2016.01.038

[63] Junming Xu. 2001. *Topological Structure and Analysis of Interconnection Networks*. Springer US. https://doi.org/10.1007/978-1-4757-3387-7

[64] Hao Yin, Austin R. Benson, and Jure Leskovec. 2018. Higher-order clustering in networks. *Physical Review E* 97, 5 (May 2018), 052306. https://doi.org/10.1103/physreve.97.052306

[65] Hao Yin, Austin R. Benson, and Jure Leskovec. 2019. The Local Closure Coefficient: A New Perspective On Network Clustering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19)*. ACM. https://doi.org/10.1145/3289600.3290991

[66] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. 2017. Local Higher-Order Graph Clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. ACM. https://doi.org/10.1145/3097983.3098069

[67] M. Yuan. 2024. Central limit theorem for the average closure coefficient. *Acta Mathematica Hungarica* 172, 2 (March 2024), 543–569. https://doi.org/10.1007/s10474-024-01416-z

[68] Chi Zhang, Wenkai Xiang, Xingzhi Guo, Baojian Zhou, and Deqing Yang. 2023. SubAnom: Efficient Subgraph Anomaly Detection Framework over Dynamic Graphs. In *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 1178–1185. https://doi.org/10.1109/icdmw60847.2023.00154

[69] Hao Zhang, Yuanyuan Zhu, Lu Qin, Hong Cheng, and Jeffrey Xu Yu. 2017. *Efficient Local Clustering Coefficient Estimation in Massive Graphs*. Springer International Publishing, 371–386. https://doi.org/10.1007/978-3-319-55699-4_23

[70] Kangfei Zhao, Jeffrey Xu Yu, Hao Zhang, Qiyan Li, and Yu Rong. 2021. A Learned Sketch for Subgraph Counting. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD/PODS '21)*. ACM. https://doi.org/10.1145/3448016.3457289