SQUASH: Distributed Square Estimation with Provable Error Bounds for Dynamic Graphs

Shaikh Arifuzzaman
Department of Computer Science
University of Nevada, Las Vegas
Las Vegas, Nevada, USA
shaikh.arifuzzaman@unlv.edu

Shubhashish Kar
Department of Computer Science
University of Nevada, Las Vegas
Las Vegas, Nevada, USA
kars1@unlv.nevada.edu

Abstract

Counting subgraph motifs is a fundamental kernel in large-scale graph analytics, yet square (4-cycle) estimation in dynamic graphs remains challenging due to combinatorial explosion and distributed state maintenance costs. We introduce SQUASH (SQUare Approximation via Sampling and High-performance communication), a distributed framework for streaming square estimation, and demonstrate both theoretically and empirically that the average error in square counting decreases inversely with the number of workers. The results indicate that utilizing approximately 1.5% of memory compared to the exact counterpart yields a relative error of about 1%. Additionally, it provides unbiased estimates with (ϵ, δ)-approximation guarantees. By leveraging Bernstein-type concentration, we prove that $O(\frac{1}{p\epsilon^2} \log \frac{1}{\delta})$ samples are sufficient for accurate estimation, where p is the closure probability of centered 3-path. For large-scale graphs in static settings, centered 3-path sampling reduces computation time by approximately three orders of magnitude relative to exact counting method having relative error of approximately 1%. Furthermore, we provide a rigorous communication analysis showing that volume is governed by the interplay between wedge generation rate and pair-space size, achieving sublinear growth with provable phase transitions. This exposes a principled three-way trade-off between estimation accuracy, memory, and communication cost. Experiments on graphs with up to $117M$ edges demonstrate that SQUASH maintains high accuracy while establishing a mathematically grounded solution for higher-order motif estimation at scale.

CCS Concepts

• **Theory of computation → Dynamic graph algorithms**; • **Computing methodologies → Parallel algorithms**.

Keywords

Graph algorithms, approximation algorithms, dynamic graphs, square counting, provable error bounds, parallel algorithms

ACM Reference Format:
Shaikh Arifuzzaman and Shubhashish Kar. 2026. SQUASH: Distributed Square Estimation with Provable Error Bounds for Dynamic Graphs. In *Proceedings of the 55th International Conference on Parallel Processing (ICPP '26), September 28-October 01, 2026, Singapore, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3832810.3832917

ICPP '26, Singapore, Singapore

ACM ISBN 979-8-4007-2657-6/2026/09
https://doi.org/10.1145/3832810.3832917

1 Introduction

A graph, as a non-linear data structure, models a wide range of real-world systems across network science, bioinformatics, and social networks [1–6, 15, 17]. Modern networks are large-scale and dynamic, with edges continuously evolving over time. Analyzing higher-order structures is crucial for understanding properties such as community structure and interaction dynamics. Graph motifs, as small recurring subgraphs, play a fundamental role in network analysis. While triangles have been extensively studied for applications such as clustering coefficient (CC), transitivity, and triangular connectivity, squares (4-cycles) capture richer structural interactions and reveal higher-order relationships within complex networks. Consequently, square counting provides valuable insights into the underlying structural organization of these networks. Moreover, the square is the smallest non-trivial closed motif in bipartite graphs. In contrast, square counting in unipartite graphs is computationally more demanding due to the substantially larger number of candidate subgraphs.

Counting squares at scale is computationally challenging due to combinatorial explosion, making exact enumeration prohibitively expensive. Higher-order motifs also incur significantly greater intermediate memory overhead than lower-order ones. These challenges are exacerbated in distributed settings, where partitioning introduces communication and synchronization costs, along with redundant storage of neighborhood information. The problem is further complicated in *fully dynamic* graphs, where maintaining exact counts under insertions and deletions requires costly recomputation or complex state management.

To address these challenges, approximate algorithms [1, 13, 24, 30] have emerged as a practical alternative, offering significant computational savings while providing controlled error guarantees. However, existing work on approximate motif counting is largely limited to static graphs, sequential settings, or lower-order motifs such as triangles. In particular, there is a lack of principled approaches for higher-order motif estimation–such as squares–in fully dynamic graphs under distributed execution, where both statistical accuracy and communication efficiency must be jointly addressed.

We address this gap by developing a distributed framework for square counting in fully dynamic graph streams, with a focus on provably accurate approximation. Our method employs a randomized pairing mechanism for unbiased estimation under arbitrary updates, supported by rigorous variance and error bounds and explicit communication analysis.

To the best of our knowledge, this work presents the first distributed algorithm for square (4-cycle) estimation in fully dynamic graphs with provable accuracy guarantees. Our formulation unifies dynamic graph processing, higher-order motif estimation, and distributed systems considerations into a single principled framework.

Contributions. The key contributions of this paper are as follows:

- **First provably accurate distributed square estimation in fully dynamic graphs.** We design a random-pairing based estimator tailored for distributed environments for 4-cycle counting that operates under arbitrary edge insertions and deletions. The estimator produces unbiased estimates and is accompanied by provable error bounds derived via concentration inequalities, establishing a principled foundation for higher-order motif estimation in dynamic distributed settings.
- **Theoretical characterization of accuracy and communication trade-offs.** We provide formal analysis of estimator variance and error bounds for the sampling techniques, alongside theoretical communication cost analysis for distributed execution of the wedge-based framework.
- **Comprehensive evaluation on static and fully dynamic graphs.** We conduct extensive experiments on large real-world graphs of different domains using distributed-memory systems. We further analyze memory efficiency and robustness under varying update patterns.

2 Preliminaries

We consider a simple, undirected, unweighted graph $G = (V, E)$, where V denotes the set of vertices and $E \subseteq V \times V$ denotes the set of edges. The graph contains no self-loops or multi-edges. Exact motif counts in many large-scale applications are computationally prohibitive and therefore, approximate counts with provable guarantees are sufficient. For parameters $\epsilon, \delta \in (0, 1)$, an (ϵ, δ)-approximation of a quantity x is a random variable $\hat{x}$ such that

$$\Pr\left(|\hat{x} - x| > \epsilon x\right) \leq \delta.$$

A square (4-cycle) is defined as a set of four distinct vertices $\{u, v, x, y\}$ such that the edges (u, v), (v, x), (x, y), and (y, u) exist in E_t. We emphasize that we count *non-induced* squares, i.e., additional edges among these vertices do not invalidate the count. In this work, we focus on estimating the number of non-induced squares using sampling-based techniques that provide such probabilistic guarantees. Along with static networks, we study the problem in the context of *fully dynamic graph streams*. Let $\{e_1, e_2, \ldots\}$ denote a sequence of edge updates, where each update $e_i = (u, v, \sigma)$ consists of an edge (u, v) and an operation $\sigma \in \{+, -\}$ indicating insertion or deletion, respectively. At time t, the graph is defined as $G_t = (V_t, E_t)$, where E_t is the set of edges after processing the first t updates, and V_t is the corresponding set of vertices. Table 1 lists the notations used in this paper.

Table 1: Summary of Notations Used in this Paper.

Symbol	Description
$G^{(t)} = (V^{(t)}, E^{(t)})$	Graph at time t
$n^{(t)}, m^{(t)}$	#vertices, #edges
$\Pi = (u, v, \delta)$	Dynamic stream, $\delta \in \{\pm 1\}$
$S^{(t)}$	Square count
$\hat{S}^{(t)}$	Sequential estimate
$c_i^{(t)}, \bar{c}^{(t)}$	Worker and global estimates
$R^{(t)}, R_i^{(t)}$	Reservoir (global / worker)
k	Reservoir size
$T^{(t)}, y^{(t)}$	Effective stream and sample size
p	Number of Partitions / Probability
$p_3^{(t)}$	3-edge sampling probability
l_{cp}	#Centered 3-path in graph
$\lambda^{(t)}$	Inverse weight $(1/p_3^{(t)})$
$p_r^{(t)}$	r-edge sampling probability
$N(u), N_R(u)$	Graph / reservoir neighbors
$c_\ell^{(t)}$	Overlap counts ($\ell = 0, 1, 2$)
$T, \lvert T\rvert$	Worker set and size
$M = \lvert T\rvert k$	Total memory

3 Related Work

Motif counting is fundamental in large-scale graph analytics, with applications in network science, biology, and social systems. While triangle counting is well studied [1, 31, 34], extending to higher-order motifs such as squares (4-cycles) is more challenging due to combinatorial explosion and stronger edge dependencies.

Exact and Distributed Square Counting. In sequential settings, several works have explored exact counting of small subgraphs and graphlets, including higher-order motifs [23, 25, 27]. These approaches achieve efficiency through combinatorial pruning and ordering techniques but are fundamentally limited by memory and computational constraints at scale. In distributed settings, relatively few works address square counting directly. The work [32] transforms the input graph into a degree-ordered directed graph to reduce redundant enumeration and distributes the computation across nodes. While effective, such approaches focus on exact counting in static graphs and incur significant communication overhead in distributed environments.

Butterfly and Bipartite Motif Counting. Squares are closely related to butterflies (i.e., 2×2 bicliques) in bipartite graphs, and a large body of work has focused on efficient butterfly counting. Tang et al. [33] propose MONARCH, which reduces computational and communication overhead by leveraging only first-hop neighborhood information. Sanei et al. [28] introduce a layer-based priority technique that minimizes $\sum_{v \in S} \deg(v)^2$, while Wang et al. [35] propose a vertex-priority method (BFC-VP) that reduces complexity to $O\left(\sum_{(u,v)\in E} \min\{\deg(u), \deg(v)\}\right)$. These methods do not address fully dynamic updates in distributed environments.

Approximate and Sampling-Based Methods. Approximation techniques have been widely studied to address scalability in motif counting. Prior work [13] includes path sampling for 4-cycles and sparsification methods such as DOULION [34] and wedge sampling [30] for triangles. However, extending these approaches to 4-cycles is non-trivial due to higher-order edge dependencies, complicating

Table 2: Comparison with prior work. Tri: triangle, 4C: 4-cycle, Bfly: butterfly. (✓: supported, ✗: not supported, ∼: partial)

Work	Motif	Dyn	Dist	Approx	Prov
Jha et al. [13]	4C	✗	✗	✓	∼
DOULION [34]	Tri	✗	✗	✓	✓
Wedge [30]	Tri	✗	✗	✓	✓
TRIÈST [31]	Tri	✓	✗	✓	✓
ABACUS [24]	Bfly	✓	∼	✓	∼
MONARCH [33]	Bfly	✗	✓	✗	✗
Steil et al. [32]	4C	✗	✓	✗	✗
This Work	4C	✓	✓	✓	✓

unbiased estimation and variance control. Consequently, principled sampling methods for higher-order motifs remain relatively underexplored, especially in distributed settings.

Dynamic Graph Streams. Dynamic graph algorithms have been extensively studied, particularly for triangle counting in streaming settings (e.g., TRIEST [31] and sketch-based methods [21]). However, these approaches do not generalize well to 4-cycles due to higher combinatorial complexity. The studies [22] and [19] present approximation algorithms for counting 4-cycles in insertion-only graph streams. Recent works such as ABACUS [24] and PARABACUS address dynamic bipartite motifs, but existing methods remain limited in scope, lacking distributed scalability or unified theoretical guarantees. General-purpose distributed frameworks such as Pregel [20], PowerGraph [9], and GraphX [10] support scalable graph analytics but offer limited capability for higher-order motif estimation under dynamic updates with provable guarantees.

Summary and Positioning. Prior work addresses approximation, dynamic updates, or distributed execution individually, but not all three for higher-order motifs such as squares. To the best of our knowledge, this work is the first to provide a unified, communication-efficient distributed framework for square (4-cycle) estimation in fully dynamic graphs with provable guarantees.
Table 2 highlights that prior work addresses only subsets of the problem dimensions, whereas our approach uniquely combines fully **dyn**amic support, **dist**ributed execution, and **prov**able accuracy for square **approx**imation.

4 Distributed Exact Square Counting

In this section, we discuss the challenges of exact square counting in static and dynamic networks. We also analyze the communication cost for wedge-based framework in distributed settings.

4.1 Challenges in Static & Dynamic Networks

Efficient square counting in distributed-memory systems is fundamentally constrained by two interacting factors: *communication overhead* and *load imbalance*. When a graph is partitioned across multiple processes, edges are categorized as *interior edges*, whose endpoints lie within the same partition, and *cut edges*, which connect vertices across partitions. Partitioning tools such as METIS

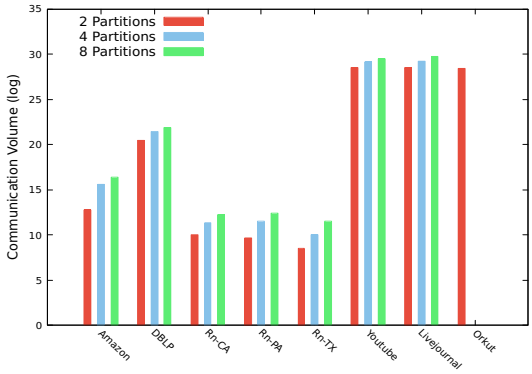


Figure 1: Communication volume (log) for exact distributed square counting (METIS partitioning); the communication volume increases with the number of partitions.

[16], ParMETIS, JET [8], aim to minimize the number of cut edges while maintaining balanced partition sizes, thereby reducing inter-process communication. However, minimizing edge cuts alone does not guarantee balanced computational workload [1]. The highly skewed degree distributions of real-world social networks make achieving balanced graph partitioning a challenging task. Figure 1 shows the increase in communication volume with the increased number of partitions regarding exact square counting.
In square counting, the computational cost is driven by wedge enumeration, i.e., the number of neighbor pairs processed at each vertex. Since this cost scales with vertex degree, skewed degree distributions can lead to substantial imbalance even when partitions are balanced in size. As a result, some processes may incur significantly higher computational load than others, causing synchronization bottlenecks during global aggregation. In practice, the overall runtime is dictated by the slowest process, making workload imbalance a critical performance limiter. While effective for static graphs, this approach becomes prohibitively expensive in fully dynamic settings, where the graph evolves continuously through edge insertions and deletions. Additionally, each update can affect a potentially large number of wedges and, consequently, multiple squares.
A further challenge arises from the distributed nature of square formation. A square may span up to four distinct partitions, requiring multi-hop communication to identify all the squares. This leads to increased communication volume and synchronization overhead. In dynamic settings, frequent updates may trigger repeated communication, making it critical to minimize both the volume and frequency of synchronization. Conversely, replicating neighborhood information locally can reduce communication but increases memory usage. For dynamic updates, partition imbalance can worsen over time due to structural drift, requiring robust update handling that avoids global repartitioning.

4.2 Distributed Algorithm for Static Graphs

Algorithm 1 presents a distributed wedge-based formulation for exact square counting. The input graph $G = (V, E)$ is partitioned into p disjoint subgraphs $P_1, P_2, \ldots, P_p$, each assigned to a distinct process.

Algorithm 1: Distributed Exact Square Counting in Static Graphs

Input: Graph $G = (V, E)$, number of partitions p
Output: Total number of squares S

```
Partition G into {P_1, ..., P_p};
Initialize wedge_count_local ← {};
foreach v ∈ V_local do
    foreach {u, w} ∈ (N(v) choose 2) do
        wedge_count_local(u, w) ←
          wedge_count_local(u, w) + 1;
wedge_count_global ← GatherAndReduce(wedge_count_local);
S ← 0;
foreach (u, w) in wedge_count_global do
    S ← S + (wedge_count_global(u,w) choose 2);
return S/2;
```

Each process operates on its local subgraph $G_{\text{local}} = (V_{\text{local}}, E_{\text{local}})$, where $G_{\text{local}} \subseteq G$.

The key idea is to count *wedges* (i.e., length-2 paths) centered at each local vertex and aggregate these counts globally to recover the number of squares. For each vertex $v \in V_{\text{local}}$, we enumerate all unordered pairs of neighbors $(u, w) \in \binom{N(v)}{2}$ and maintain a local count of such wedge pairs using a hash map. Intuitively, each square corresponds to two wedges sharing the same endpoint pair; thus, once global wedge counts are aggregated, the total number of squares can be computed combinatorially.

After local computation, processes collectively perform a global aggregation (e.g., via `MPI_Gather` and aggregation) to obtain the $wedge\_count_{\text{global}}$. The final square count is computed by summing $\binom{wedge\_count_{\text{global}}(u,w)}{2}$ over all vertex pairs. This formulation avoids explicit enumeration of 4-cycles and instead relies on efficient aggregation of local statistics.

The local computation cost is $O\left(\sum_{v \in V_{\text{local}}} \deg(v)^2\right)$, while communication cost depends on the number of distinct vertex pairs shared across partitions.

4.3 Communication Cost for Wedge-Based Framework

In distributed square counting, communication arises from aggregating wedge counts across partitions. We analyze the expected communication volume by modeling the number of *unique vertex pairs* contributed by each partition during the global aggregation phase.

Preliminaries. Let V_l and V_g denote the sets of local and ghost vertices (owned by other partition) in a partition, respectively, and let $V_p = V_l \cup V_g$ be the set of vertices stored in that partition. We denote their sizes as $n_l = |V_l|$, $n_g = |V_g|$, and $n_p = |V_p|$. Let d_v denote the degree of vertex v. Each partition generates wedges centered at its local vertices. The total number of wedges generated in a partition is:

$$n_{wp} = \sum_{v \in V_l} \binom{d_v}{2} = \frac{1}{2}\left(\sum_{v \in V_l} d_v^2 - \sum_{v \in V_l} d_v\right). \tag{1}$$

Each wedge contributes a pair of endpoints. The universe of possible endpoint pairs within a partition is:

$$n_{pp} = \binom{n_p}{2}. \tag{2}$$

Expected Number of Unique Pairs. We model wedge endpoint generation as sampling n_{wp} pairs uniformly (with replacement) from the space of n_{pp} possible pairs.[1] Let X denote the number of *distinct* pairs observed. Using indicator variables :

$$\mathbb{E}[X] = n_{pp}\left(1 - \left(1 - \frac{1}{n_{pp}}\right)^{n_{wp}}\right). \tag{3}$$

This expression characterizes the expected number of unique pairs contributed by a partition to the global aggregation, and thus directly models its communication footprint.

Closed-Form Approximation. For large n_{pp}, we use the approximation $(1 - \frac{1}{n_{pp}})^{n_{wp}} \approx e^{-n_{wp}/n_{pp}}$, yielding:

$$\mathbb{E}[X] \approx n_{pp}\left(1 - e^{-n_{wp}/n_{pp}}\right). \tag{4}$$

Using a Taylor expansion, we further obtain:

$$\mathbb{E}[X] \approx n_{wp} - \frac{n_{wp}^2}{2n_{pp}} + O\left(\frac{n_{wp}^3}{n_{pp}^2}\right). \tag{5}$$

Birthday Paradox Interpretation. Equation 5 admits a natural interpretation via the birthday paradox [14]. The expected number of pair collisions is:

$$\mathbb{E}[\text{collisions}] = \binom{n_{wp}}{2} \cdot \frac{1}{n_{pp}}. \tag{6}$$

Thus, the number of unique pairs is:

$$\mathbb{E}[X] \approx n_{wp} - \frac{n_{wp}(n_{wp} - 1)}{2n_{pp}}, \tag{7}$$

which corresponds to the first two terms of Equation 5, where for large graphs, $n_{wp} \gg 1$. This highlights that communication is reduced by collisions among wedges mapping to the same endpoint pair.

Total Communication Across Partitions. Using linearity of expectation, for p partitions, the total expected communication volume is:

$$\mathbb{E}[X_{\text{total}}] = \sum_{i=1}^{p} n_{pp}^{(i)}\left(1 - \left(1 - \frac{1}{n_{pp}^{(i)}}\right)^{n_{wp}^{(i)}}\right). \tag{8}$$

[1] This approximation captures the average-case behavior and becomes increasingly accurate for large graphs with moderate degree skew.

Key Insights. The analysis reveals several important properties:

- **Sublinear communication growth.** When $n_{wp} \ll n_{pp}$, we have $\mathbb{E}[X] \approx n_{wp}$, implying minimal collision and near-linear communication. However, as n_{wp} grows relative to n_{pp}, collisions increase, reducing the number of unique pairs and thus communication volume. When $n_{wp} = \Theta(n_{pp})$, the expected communication saturates at $\Theta(n_{pp})$, indicating a phase transition from sparse to dense collision regimes.
- **Impact of partition size.** Larger n_p increases n_{pp}, which reduces collision probability and increases communication. Thus, overly large partitions may increase communication despite reducing edge cuts.
- **Role of degree distribution.** Since $n_{wp} \propto \sum d_v^2$, high-degree vertices dominate communication cost, highlighting the importance of degree-aware partitioning.
- **Communication–computation trade-off.** Increasing local redundancy (larger V_g) increases n_p and n_{pp}, reducing computation duplication but potentially increasing communication.

Implication. This analysis shows that communication cost is governed not only by edge cuts, but by the interplay between *wedge generation rate* (n_{wp}) and *pair space size* (n_{pp}). This provides a principled basis for designing communication-efficient distributed motif algorithms.

Concentration Bound. Let X_i denote the number of unique pairs in partition i, and $S = \sum_{i=1}^{p} X_i$. Since $1 \le X_i \le n_{wp}^{(i)}$, Hoeffding's inequality [12] gives:

$$\Pr\left(|S - \mathbb{E}[S]| \ge \epsilon\right) \le 2\exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^{p}(n_{wp}^{(i)} - 1)^2}\right). \tag{9}$$

Optimization. The exact square counting approach incurs substantial communication overhead in distributed wedge-based framework, with the communication volume increasing proportionally to the graph size. To mitigate this limitation, Algorithm 1 can be optimized by exploiting partition locality. Specifically, if both endpoints of a wedge reside in the same partition, the corresponding computation can be performed entirely within that partition, eliminating the need for inter-partition communication. Consequently, only wedge-endpoint pairs spanning multiple partitions require global processing. Each worker therefore transmits only the cross-partition wedge-endpoint pairs together with the locally computed square count corresponding to wedges whose endpoints lie within the same partition.

For large-scale graphs, the majority of wedge-endpoint pairs are typically confined to the same partition. As a result, this optimization substantially reduces the communication volume by avoiding the transmission of locally computable wedge-endpoint pairs, thereby improving the scalability and communication efficiency of the distributed algorithm.

5 Approximation Algorithms for Static Graphs

This section is centered around the edge, wedge, and centered 3-path sampling techniques and their provable error bounds.

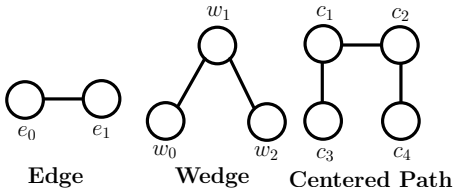


Figure 2: Constituent blocks of a square: edge, wedge, and centered-path.

5.1 Edge-Based Approximation

Let the squares in the graph are numbered from 1 to S. Assuming the number of squares containing edge e is X. X_i is 1 if i-th square contains e, otherwise 0. $X = \sum_{i=1}^{S} X_i$.

$$E[X] = \sum_{i=1}^{S} E[X_i] = \sum_{i=1}^{S} P[X_i = 1]$$

Since every square has four edges and m is the number of edges in the graph. $P[X_i = 1] = \frac{4}{m}$. Therefore,

$$E[X] = S \times \frac{4}{m} \implies S = E[X] \times \frac{m}{4}.$$

Variance Analysis. The approximate square count is $\hat{S}$

$$\mathrm{Var}(\hat{S}) = \mathrm{Var}\left[\frac{m}{4}\sum_{i=1}^{S} X_i\right] = \frac{m^2}{16}\left[\sum_{i=1}^{S}\mathrm{Var}[X_i] + 2\sum_{i<j}\mathrm{Cov}(X_i, X_j)\right]. \tag{10}$$

For edge sampling, a pair of squares can share one edge or one wedge. Taking this into consideration and let l_E be the number of squares that share at least one edge,

$$\mathrm{Var}(\hat{S}) \le \frac{m}{4}\left(S + l_E\right). \tag{11}$$

Applying Chebyshev's inequality [29] to derive error bounds,

$$\Pr\left(|\hat{S} - S| \ge \epsilon S\right) \le \left(\frac{m(S + l_E)}{4\epsilon^2 S^2}\right).$$

5.2 Wedge-Based Approximation

To obtain a scalable alternative, we adopt a wedge-based randomized estimator that avoids exhaustive enumeration while preserving provable accuracy guarantees.

Let $G = (V, E)$ be a simple undirected graph. Define the total number of wedges in G as

$$W = \sum_{v \in V}\binom{d_v}{2}, \tag{12}$$

Let, the squares in the graph are numbered from 0 to S. Assuming the number of squares containing wedge w is $X = S_w$. Similar to Edge-Based approximation, we can show that, $S = E[X] \times \frac{W}{4}$.

For wedge sampling, a pair of squares can share at most one wedge. Let l_W be the number of squares that share one wedge and similar to Equation 11,

$$\mathrm{Var}(\hat{S}) \le \frac{W}{4}\left(S + l_W\right). \tag{13}$$

Applying Chebyshev's inequality to derive error bounds,

$$\Pr\left(|\hat{S} - S| \geq \epsilon S\right) \leq \left(\frac{W(S + l_W)}{4\epsilon^2 S^2}\right).$$

5.3 Centered 3-Path Based Approximation

Let $G = (V, E)$ be a simple undirected graph. Figure 2 shows the centered 3-path containing edges (c_3, c_1), (c_1, c_2), and (c_2, c_4), where $c_3 > c_2$ and $c_4 > c_1$. Define the total number of centered 3-paths in G as

$$l_{cp} = \sum_e \lambda_e, \tag{14}$$

where $\lambda_e = L_{u,v}L_{v,u}$. And $L_{u,v}$ is the number of neighbors of u greater than v. Let, S denote the total number of squares in G. Since each square contains exactly one centered 3-path, the probability that a uniformly sampled centered 3-path belongs to a square is

$$p = \frac{S}{l_{cp}}. \tag{15}$$

We sample s centered 3-paths independently and uniformly at random with replacement. For the i-th sampled centered 3-path, define the indicator variable $X_i = 1$ if the sampled centered 3-path participates in one square and otherwise $X_i = 0$. Let

$$\hat{p} = \frac{1}{s}\sum_{i=1}^{s} X_i. \tag{16}$$

The centered 3-path based estimator of the global square count is then

$$\hat{S} = l_{cp}\hat{p} = \frac{l_{cp}}{s}\sum_{i=1}^{s} X_i. \tag{17}$$

Theorem 1 (Unbiasedness). *The estimator $\hat{S}$ in Equation (17) is unbiased: $\mathbb{E}[\hat{S}] = S$.*

Proof. Since each sampled centered 3-path is drawn uniformly at random, $\mathbb{E}[X_i] = p = \frac{S}{l_{cp}}$. Therefore,

$$\mathbb{E}[\hat{S}] = \frac{l_{cp}}{s}\sum_{i=1}^{s}\mathbb{E}[X_i] = \frac{l_{cp}}{s}\cdot s \cdot \frac{S}{l_{cp}} = S.$$

□

Theorem 2 (Variance). *The variance of $\hat{S}$ is*

$$\mathrm{Var}(\hat{S}) = \frac{l_{cp}^2}{s}p(1-p) = \frac{l_{cp}S}{s}\left(1 - \frac{S}{l_{cp}}\right). \tag{18}$$

Consequently, the relative standard error satisfies

$$\mathrm{RSE}(\hat{S}) = \frac{\sqrt{\mathrm{Var}(\hat{S})}}{S} = \sqrt{\frac{1-p}{sp}}. \tag{19}$$

Proof. Because the centered 3-paths are sampled independently with replacement, the variables X_i are i.i.d. Bernoulli(p). Hence,

$$\mathrm{Var}\left(\sum_{i=1}^{s} X_i\right) = s\,p(1-p).$$

Applying the scaling in Equation (17),

$$\mathrm{Var}(\hat{S}) = \left(\frac{l_{cp}}{s}\right)^2 s\,p(1-p) = \frac{l_{cp}^2}{s}p(1-p).$$

Substituting $p = \frac{S}{l_{cp}}$ yields Equation (18). Then dividing by S^2 and taking square roots gives Equation (19). □

Equation (19) is particularly informative: it shows that the estimator accuracy is governed primarily by the centered 3-path closure probability p, rather than directly by the graph size. In particular, when square-supporting centered 3-paths are not extremely rare, the required number of samples remains moderate even for massive graphs. To achieve relative standard error at most τ, it suffices that

$$s \geq \frac{1-p}{p\tau^2}. \tag{20}$$

This dependence on p rather than $|V|$ or $|E|$ makes the estimator practically attractive for large graphs with non-negligible square density among centered 3-paths.

Theorem 3 (High-Probability Accuracy). *For any $\epsilon, \delta \in (0, 1)$,*

$$\Pr\left(|\hat{S} - S| \geq \epsilon S\right) \leq 2\exp\left(-\frac{s\epsilon^2 p^2}{2p(1-p) + \frac{2}{3}\epsilon p}\right). \tag{21}$$

Equivalently, it suffices to choose

$$s = O\left(\frac{1}{p\epsilon^2}\log\frac{1}{\delta}\right) \tag{22}$$

to obtain an (ϵ, δ)-approximation.

Proof. Since $X_i \in [0, 1]$ are independent Bernoulli random variables, a Bernstein-type concentration bound for bounded independent variables applies to $\hat{p} = \frac{1}{s}\sum_i X_i$. Specifically,

$$\Pr(|\hat{p} - p| \geq \eta) \leq 2\exp\left(-\frac{s\eta^2}{2p(1-p) + \frac{2}{3}\eta}\right).$$

Because $\hat{S} = l_{cp}\hat{p}$ and $S = l_{cp}p$, the event $|\hat{S} - S| \geq \epsilon S$ is equivalent to $|\hat{p} - p| \geq \epsilon p$. Substituting $\eta = \epsilon p$ yields Equation (21). Rearranging the exponent gives the sufficient sample complexity in Equation (22). □

Similarly, for other sampling techniques, Chebyshev's inequality can be applied to derive a (generally loose) upper bound on the probability of large deviations,

$$\Pr\left(|\hat{S} - S| \geq \epsilon S\right) \leq \left(\frac{l_{cp}S - S^2}{s\epsilon^2 S^2}\right).$$

Distributed Estimation. Let the graph be partitioned into p subgraphs $P_1, \ldots, P_p$. Consequently, the edges are partitioned into p partitions, $E = E_1 \cup \ldots \cup E_p$. For partition P_i, the local square estimator is $\hat{S}_i$ and the global estimator is

$$\hat{S} = \sum_{i=1}^{p}\hat{S}_i. \tag{23}$$

Theorem 4 (Distributed Unbiasedness). *If each partition samples centered 3-paths uniformly from its local centered 3-paths, then the estimator in Equation (23) is unbiased: $\mathbb{E}[\hat{S}] = S$.*

Proof. In the distributed setting, each edge e is assigned to a unique partition E_i. Although an edge may participate in multiple squares, every square contains exactly one centered 3-path. Consequently,

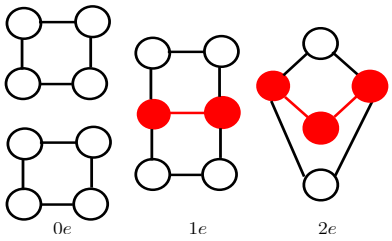


Figure 3: Edge overlap between two squares: sharing 0, 1, or 2 edges.

each square is enumerated only by the partition that owns its centered 3-path, ensuring that every square is counted exactly once without over-counting.

For each partition, the same argument as in the first theorem gives $\mathbb{E}[\hat{S}_i] = S_i$, where S_i denotes the contribution associated with centered 3-paths in partition P_i. Summing across p partitions and using linearity of expectation yields

$$\mathbb{E}[\hat{S}] = \sum_{i=1}^{p} \mathbb{E}[\hat{S}_i] = \sum_{i=1}^{p} S_i = S.$$

□

6 Sequential Approximation for Fully Dynamic Graphs

In fully dynamic graph streams, where edges both arrive and depart, storing the entire graph or recomputing higher-order structures is infeasible. We therefore propose a one-pass, memory-bounded estimator that maintains a uniform edge sample, combining random pairing with inverse-probability estimation to ensure provably accurate counting under insertions and deletions.

Sampling Model. Let $G^{(t)} = (V^{(t)}, E^{(t)})$ denote the graph at time t, and let $m^{(t)} = |E^{(t)}|$. The algorithm maintains a reservoir $S^{(t)}$ of at most k edges using standard reservoir sampling extended to handle deletions via compensation counters. We assume $k \ll m^{(t)}$, which is typical in streaming settings. A square is detected when a final edge completes a 3-edge path already present in the sample.

Theorem 5 (Uniform Triplet Sampling). *At any time t, any fixed set of three distinct edges is present in the reservoir with probability $p_3^{(t)}$ where*

$$p_3^{(t)} = \frac{\binom{k}{3}}{\binom{m^{(t)}}{3}} \approx \left(\frac{k}{m^{(t)}}\right)^3. \tag{24}$$

Proof. The result follows directly from uniform sampling without replacement in reservoir sampling. The joint inclusion probability of any fixed set of edges is determined by combinatorial selection, yielding Equation (24). □

Estimator. Let $\lambda^{(t)} = \frac{1}{p_3^{(t)}}$ be the inverse-probability weight. For each update (u, v, δ):

- If $\delta = +$ (insertion), and (u, v) completes a square using sampled edges, we increment the estimate by $\lambda^{(t-1)}$.
- If $\delta = -$ (deletion), and (u, v) destroys a previously counted square, we decrement the estimate by $\lambda^{(t-1)}$.

Let $\hat{S}^{(t)}$ denote the cumulative estimate.

Theorem 6 (Unbiasedness). *At any time t, $\mathbb{E}[\hat{S}^{(t)}] = S^{(t)}$*

Proof. The probability that its supporting 3 edges are present is $p_3^{(t)}$, and the estimator applies weight $1/p_3^{(t)}$. Hence, the expected contribution is 1 (or -1 for deletion). Summing over all squares yields the result. □

Variance Analysis. Let $X_i^{(t)}$ indicate whether square i is observed. Then

$$\hat{S}^{(t)} = \lambda^{(t)} \sum_i X_i^{(t)}.$$

Theorem 7 (Variance). *The variance satisfies*

$$\mathrm{Var}(\hat{S}^{(t)}) = \frac{S^{(t)}(1 - p_3^{(t)})}{p_3^{(t)}} + 2 \sum_{i<j} \left(\frac{p_{ij}^{(t)}}{(p_3^{(t)})^2} - 1 \right), \tag{25}$$

where $p_{ij}^{(t)}$ denotes the joint sampling probability of squares i and j.

Proof. The result follows from expanding $\mathrm{Var}(\lambda \sum_i X_i)$ and applying variance decomposition. The first term captures individual contributions, while the second accounts for dependencies.

Dependency Structure of Overlapping Squares. Dependencies arise when squares share edges. Let two squares share $\ell \in \{0, 1, 2\}$ edges. Then:

- $\ell = 0$: 8 distinct edges,
- $\ell = 1$: 7 distinct edges,
- $\ell = 2$: 6 distinct edges.

Let

$$p_r^{(t)} = \frac{\binom{k}{r}}{\binom{m^{(t)}}{r}} \approx \left(\frac{k}{m^{(t)}}\right)^r.$$

$$\mathrm{Cov}(X_i, X_j) = p_{r_\ell}^{(t)} - (p_3^{(t)})^2. \tag{26}$$

Using approximation:

$$\frac{p_{r_\ell}^{(t)}}{(p_3^{(t)})^2} \approx \left(\frac{k}{m^{(t)}}\right)^{r_\ell - 6}. \tag{27}$$

Thus:

- $\ell = 2$: $O(1)$ contribution (dominant),
- $\ell = 1$: $O(k/m)$,
- $\ell = 0$: $O((k/m)^2)$.

Key Insight. The dominant dependency arises from squares sharing two edges, while contributions from disjoint squares vanish as $k \ll m^{(t)}$. This locality explains why estimator variance remains controlled even in large graphs.

Closed-Form Approximation. Let

$$C_{ov}^{(t)} = O(c_2^{(t)}) + O\left(c_1^{(t)} \frac{k}{m^{(t)}}\right) + O\left(c_0^{(t)} \left(\frac{k}{m^{(t)}}\right)^2\right), \tag{28}$$

where $c_\ell^{(t)}$ counts square pairs sharing ℓ edges.
Combining terms:

$$\mathrm{Var}(\hat{S}^{(t)}) \approx S^{(t)} \left(\frac{m^{(t)}}{k}\right)^3 + C_{ov}^{(t)}. \tag{29}$$

□

High-Probability Guarantee (Bernstein Bound).

Theorem 8. *For any $\epsilon, \delta \in (0,1)$,*

$$\Pr\left(|\hat{S}^{(t)} - S^{(t)}| \geq \epsilon S^{(t)}\right) \leq 2\exp\left(-\frac{\epsilon^2 S^{(t)2}}{2\mathrm{Var}(\hat{S}^{(t)}) + \frac{2}{3}\epsilon\lambda^{(t)}S^{(t)}}\right). \tag{30}$$

Proof. This follows from Bernstein's inequality applied to bounded random variables with variance $\mathrm{Var}(\hat{S}^{(t)})$. □

Although the Bernstein bound provides tight high-probability guarantees, its dependence on variance and inverse-probability weights complicates explicit memory analysis.

Proposition 1 (Sufficient Memory Budget). *Let $\hat{S}^{(t)}$ be the sequential dynamic estimator at time t. Ignoring lower-order covariance terms due to overlapping squares, a sufficient condition for*

$$\Pr\left(\frac{|\hat{S}^{(t)} - S^{(t)}|}{S^{(t)}} \geq \epsilon\right) \leq \delta$$

is

$$k = \Omega\left(m^{(t)}\left(\frac{1}{\delta\epsilon^2 S^{(t)}}\right)^{1/3}\right).$$

Proof. Using Chebyshev's inequality together with the dominant variance term

$$\mathrm{Var}(\hat{S}^{(t)}) \lesssim S^{(t)}\left(\frac{m^{(t)}}{k}\right)^3,$$

it suffices to require

$$S^{(t)}\left(\frac{m^{(t)}}{k}\right)^3 \leq \delta\epsilon^2 (S^{(t)})^2.$$

Rearranging yields the stated condition. □

Discussion. This formulation yields several important insights:

- **Memory–accuracy trade-off:** Variance scales as $\Theta(S^{(t)}(m^{(t)}/k)^3)$, showing that increasing memory reduces error polynomially.
- **Streaming scalability:** The algorithm operates in one pass with $O(k)$ memory.
- **Local dependency structure:** Variance is dominated by locally overlapping squares.
- **Provable accuracy:** The estimator is unbiased with tight concentration guarantees.

This establishes a principled framework that is both statistically rigorous and practically scalable.

7 Distributed Approximation for Fully Dynamic Graphs

We now extend the sequential dynamic estimator introduced in Section 6 to a distributed-memory setting. The key idea is to combine independent random pairing across workers with lightweight aggregation, thereby avoiding the need to maintain a globally consistent sample and retaining the estimator's statistical guarantees.

Algorithm 2: Coordinator: Broadcast Coordination

Input: Stream $\Pi = \{(u, v, \delta)\}$, worker set T
foreach *update* $(u, v, \delta) \in \Pi$ **do**
  Broadcast (u, v, δ) to all workers $i \in T$;

Algorithm 3: Worker i: Random Pairing-Based Estimation

Input: Broadcast stream Π, memory budget k
Initialize $R_i \leftarrow \emptyset$, $m_i \leftarrow 0$, $n_{b,i}, n_{g,i} \leftarrow 0$;
Initialize global estimate $c_i \leftarrow 0$;
foreach $(u, v, \delta) \in \Pi$ **do**
  Compute $T_i = m_i + n_{b,i} + n_{g,i}$ and $y_i = \min(k, T_i)$;
  Compute $p_{3,i} = \frac{\binom{y_i}{3}}{\binom{T_i}{3}}$, $\lambda_i = 1/p_{3,i}$;
  $count \leftarrow 0$;
  foreach $w \in (N_{R_i}(u) \setminus v)$ **do**
    foreach $x \in ((N_{R_i}(w) \cap N_{R_i}(v)) \setminus u)$ **do**
      $count \leftarrow count + 1$;
  Update c_i using $\delta \cdot count \cdot \lambda_i$;
  Update $(R_i, m_i, n_{b,i}, n_{g,i})$ using fully dynamic reservoir maintenance;
  Periodically emit (i, c_i) to aggregators;

Distributed Stream Model. We adopt a broadcast stream model in which each worker observes the same sequence of updates (u, v, δ), while maintaining an independent reservoir sample. We denote the reservoir at worker i by $R_i^{(t)}$. The distributed framework consists of three logical components. A coordinator node ingests the fully dynamic stream and broadcasts updates to all workers (Algorithm 2). Each worker maintains a local reservoir and independently estimates the global square count (Algorithm 3). Aggregator node collect and combine these estimates to produce the final output (Algorithm 4). This aggregation strategy avoids the need for synchronization of sampling state across workers. Consequently, each worker produces an unbiased estimate of the global square count, and cross-worker placement of edges does not affect correctness.

Worker-Side Estimation. Each worker independently executes the dynamic reservoir-based estimator over the broadcast stream, as summarized in Algorithm 3. Here, n_b and n_g denote the compensation counters for deletions in Random Pairing Sampling [7]. Upon receiving an update, the worker computes the current sampling probability based on its effective stream size, evaluates the number of sampled 3-edge configurations that form or destroy squares, and updates its estimate using inverse-probability weighting. The reservoir is then updated using fully dynamic sampling logic that supports both insertions and deletions. Importantly, each worker maintains a complete estimator of the global square count rather than a partition-local quantity. Besides, each worker operates within a fixed memory budget k, yielding total memory $M = |T|k$.

Statistical Guarantees. We now formalize the correctness and variance properties of the distributed estimator. Let

$$\bar{c}^{(t)} = \frac{1}{|T|}\sum_{i \in T} c_i^{(t)}.$$

Algorithm 4: Aggregator: Estimate Reduction

Input: Worker outputs (i, c_i)
Initialize $\bar{c} \leftarrow 0$;
/* Runs Periodically and set $\bar{c} \leftarrow 0$ */
foreach *received* (i, c_i) **do**
  $\bar{c} \leftarrow \bar{c} + c_i/|T|$;

Theorem 9 (Unbiasedness). *The distributed estimator is unbiased, i.e.,* $\mathbb{E}[\bar{c}^{(t)}] = S^{(t)}$.

Proof. Each worker estimator is unbiased by construction (Algorithm 3). The result follows directly from linearity of expectation. □

Theorem 10 (Variance Reduction). *Assuming independence of worker reservoirs* $R_i^{(t)}$,

$$\mathrm{Var}(\bar{c}^{(t)}) = \sum_{i \in T} \mathrm{Var}\left(\frac{c_i^{(t)}}{|T|}\right) = \frac{1}{|T|^2} \sum_{i \in T} \mathrm{Var}(c_i^{(t)}).$$

Proof. The result follows from independence of the sampling processes across workers and standard properties of variance. □

The independence assumption pertains to the sampling processes rather than the graph structure. Even if a square spans edges processed on multiple machines, it remains detectable by each worker because all workers observe the same stream.

Communication Analysis. A communication performance model analytically characterizes the communication cost of a cluster using a limited set of platform-specific parameters. The Hockney communication model [11] expresses the communication cost as

$$T(m) = \alpha + m\beta.$$

where m denotes the size of the message, α represents the network latency (startup cost), and β is the reciprocal of the network bandwidth. In Algorithms 2–4, the message size is comparatively small. Therefore, latency (α) dominates bandwidth costs. For simple broadcasting for the graph stream with $E^{(t)}$ edges and T workers, the broadcasting cost can be represented as

$$T_{bcast} = |E^{(t)}||T|(\alpha + m\beta). \tag{31}$$

For small messages, MPI generally utilizes tree-structured broadcasting. Tree-structured broadcasting requires $\log_2 |T|$ communication steps to broadcast a message to all processes. The cost for tree-structured broadcasting is

$$T_{tree\_bcast} = |E^{(t)}|\lceil \log_2 |T| \rceil(\alpha + m\beta). \tag{32}$$

If every worker sends messages periodically to the coordinator for global estimation, then $(\alpha + m\beta)$ is added for each message to calculate the total communication cost. Additionally, the framework avoids global reservoir synchronization by design. As shown in Algorithms 2–4, communication consists of broadcasting updates and aggregating the messages from the workers. This significantly reduces overhead compared to approaches that maintain globally consistent samples.

Table 3: Datasets and their structural properties. Counts are reported in thousands (K), millions (M), and billions (B).

Dataset	$\|V\|$	$\|E\|$	d_{avg}	$d_{\max}$	$\|W\|$	$\|S\|$	$\|W\|/\|V\|$	$\|S\|/\|W\|$
Amazon (AZ)	334.86K	340.09K	2.03	154	2.15M	424.11K	6.45	0.19
DBLP (DB)	317.08K	782.20K	4.93	264	15.51M	34.42M	48.92	2.21
Youtube (YT)	1.13M	2.95M	5.21	28.75K	1.46B	468.29M	1287.41	0.32
Rn-CA (RoCA)	1.97M	2.75M	2.80	12	5.97M	261.57K	3.03	0.04
Rn-PA (RoPA)	1.09M	1.53M	2.82	9	3.38M	157.71K	3.10	0.04
Rn-TX (RoTX)	1.38M	1.90M	2.75	12	4.08M	181.34K	2.95	0.04
Email-Enron (EE)	36.69K	183.83K	10.02	1.38K	25.56M	36.26M	696.64	1.41
CA-HepPh (HP)	12.01K	2.66K	0.44	71	51.10K	237.64K	4.25	4.65
CA-AstroPh (AS)	18.77K	4.07K	0.43	83	37.49K	22.71K	1.99	0.60
LiveJournal (LJ)	4.00M	34.61M	17.31	14.76K	4.24B	26.37B	1060.63	6.22
Orkut (OK)	3.07M	117.19M	76.27	33.31K	45.62B	127.53B	14849.70	2.79

8 Experimental Evaluation

In this section, we assess the performance of our algorithms and techniques using a comprehensive range of datasets.

8.1 Dataset Description

We utilize a dozen real-world graph datasets from diverse domains, exhibiting varied structural properties and sizes. The real-world datasets vary considerably in scale, with the number of edges ranging from 183 thousand to more than 117 million. Key statistics are reported in Table 3. The $|S|/|W|$ ratio is consistently higher for social networks than for road networks, indicating a greater prevalence of square motifs relative to wedges in social networks. For dynamic streams, the edge ordering of the static graph is treated as the insertion sequence. We introduce deletions by randomly removing 5% and 10% of the inserted edges after their insertion. All datasets are sourced from the Stanford Network Analysis Project [18] and the Network Repository [26].

8.2 Experimental Setup & Evaluation

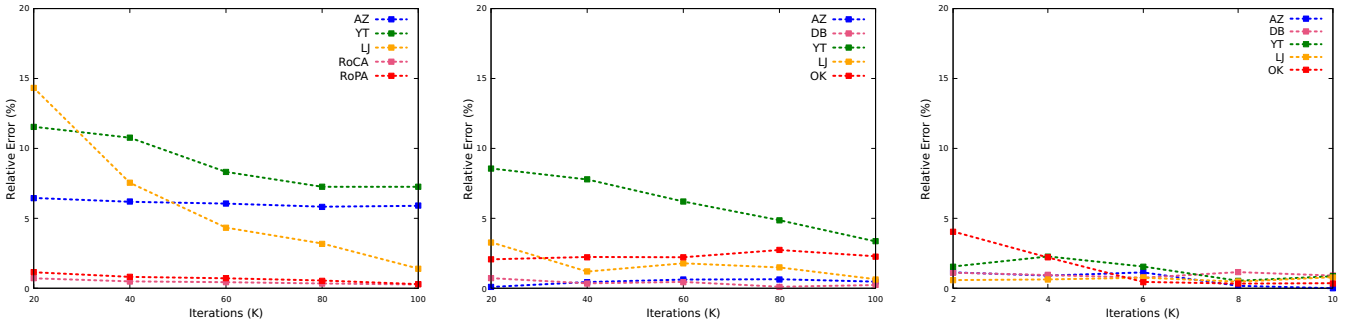
The code is compiled using g++ version 11.4.0. All experiments are conducted on a compute cluster, where each node is equipped with an Intel Xeon E5-2697v2 processor (12 cores) and 128 GB of RAM. The static networks are primarily represented using the Compressed Sparse Row (CSR) format. To adapt insertions and deletions in dynamic networks, we utilize the Packed Compressed Sparse Row (PCSR). For distributed implementation, we utilize the MPI-based framework for computation on multiple computing nodes. We evaluate the approximation techniques by relative error, which can be defined as

$$\text{Relative Error (\%)} = \frac{|C_{approx} - C_{exact}|}{C_{exact}} \times 100 \tag{33}$$

where C_{approx} and C_{exact} denote the approximate count and exact count, respectively. In this work, we report the median and average errors on three runs.

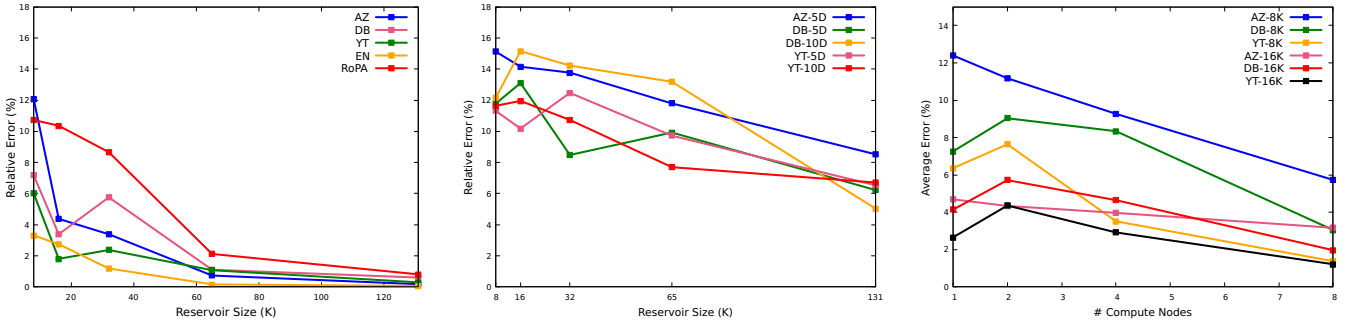
8.3 Approximate Square Count

8.3.1 ***Sampling in Static Networks.*** To estimate the square count in static graphs, we consider three sampling techniques: edge sampling, wedge sampling, and centered 3-path sampling. These three techniques involve edges of a square to estimate. Figure 4 illustrates the relative error as a function of the number of samples (edges, wedges, and centered 3-paths for the respective methods). For both edge and wedge sampling, the relative error approaches approximately 1% within 30–60K samples in social networks. In



(a) Edge Sampling (b) Wedge Sampling (c) Centered 3-Path Sampling

Figure 4: Square estimation using edge, wedge, and centered path sampling. Among the sampling techniques, Centered 3-path sampling achieves near-zero relative error with significantly fewer sampled 3-length paths than the other methods. In addition, road network datasets exhibit lower relative error compared to social networks for edge sampling technique.



(a) Insertion Only (b) Fully Dynamic (c) Distributed Sampling

Figure 5: The inverse relation between relative error and reservoir size for dynamic graphs in (a) Only-Insertion stream: All datasets exhibit decreasing error and (b) Fully dynamic network: the relative error is comparatively higher for deletion dominant streams. (c) The relationship between average error and the number of workers. The increased number of workers reduce variance, ultimately the error, leading to better accuracy with distributed scaling.

Table 4: Estimation of $\hat{p}$ using 10K samples under centered 3-path sampling.

Dataset	Sq.	C-Path	p	$\hat{p}$
Amazon	424.11K	986.36K	0.430	0.424
DBLP	34.42M	47.01M	0.732	0.724
Rn-CA	261.57K	2.03M	0.128	0.126
Rn-PA	157.71K	1.15M	0.136	0.130
Rn-TX	181.34K	1.33M	0.135	0.133
LiveJournal	26.37B	54.65B	0.483	0.472
Orkut	127.53B	948.03B	0.135	0.136
Youtube	468.29M	1.68B	0.279	0.270

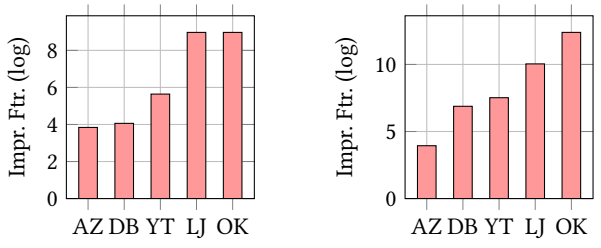
contrast, centered 3-path sampling attains a comparable error level with roughly five times fewer samples. Table 4 demonstrates the $\hat{p}$ estimation for centered 3-path sampling in different datasets. For most datasets, centered 3-path sampling achieves an accurate estimate of p within 10,000 samples.

8.3.2 ***Accuracy with Reservoir Size.*** Equation 24 shows the probability $p_3^{(t)}$'s cubic dependence on the reservoir size k in the context of square counting. Furthermore, Equation 25 establishes an inverse relationship for $Var(\hat{S}^{(t)})$. Consequently, the relative error decreases as the reservoir size increases. Figures 5a and 5b illustrate this decreasing trend in relative error with increasing reservoir size for insertion-only dynamic networks. In fully dynamic settings, higher deletion rates lead to increased relative error, while larger reservoir sizes consistently reduce it.

8.3.3 ***Error in Distributed Sampling.*** Theorem 10 establishes an inverse relationship between the variance and the number of workers in distributed settings. The accuracy of motif counting approximations is fundamentally governed by the variance of the estimator; in particular, under unbiased sampling schemes, the mean squared error reduces to the variance. In the distributed experiments shown in Figure 5c, both reservoirs having capacities of 8K and 16K edges, the plots exhibit a decreasing trend in average error. Empirically, the average error is higher with two workers and decreases as the number of workers increases.

8.4 Exact vs. Approximate Technique

Figure 6a presents a comparison of memory consumption among exact square counting [32] and distributed random pairing in dynamic setting. The evaluation measures the memory required by approximation methods to achieve a relative error of 1%. The results indicate that, for massive-scale graphs, the approximation techniques require approximately 1.5% of the memory used by exact methods to attain 1% relative error. Figure 6b demonstrates



(a) Memory usage **(b) Execution Time**

Figure 6: Comparison of exact square counting [32] and our distributed random pairing in a dynamic setting. We show improvement factors in log scale (in powers of 2, e.g., 2^{10}). (a) Memory scaling based on memory required to achieve a relative error $< 1\%$. (b) Speedup based on execution time required to achieve a relative error $< 1\%$.

the runtime improvement in centered 3-path sampling over exact counting technique. For massive-scale graphs, the sampling technique takes approximately 1000× less time compared to the exact counting method.

9 Conclusion

In this study, we analyze the communication complexity of wedge-based distributed frameworks and identified the computational bottlenecks associated with exact square enumeration. To address the limitations of exact enumeration, we present distributed algorithms for square estimation in both static and fully dynamic graphs, supported by theoretical analysis and extensive experimental evaluation. For static graphs, the centered 3-path sampling method achieves higher estimation accuracy with fewer samples than other approaches. For fully dynamic graphs, we introduce a distributed framework that employs broadcasting to propagate graph streams across multiple workers and aggregates the results from the workers. This distributed one-pass sampling framework significantly reduces the global sample synchronization cost and execution time while preserving statistical accuracy with provable theoretical guarantees.

References

[1] Shaikh Arifuzzaman, Maleq Khan, and Madhav Marathe. 2019. Fast parallel algorithms for counting and listing triangles in big graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14, 1 (2019), 1–34.

[2] D.A. Bader and K. Madduri. 2006. Parallel Algorithms for Evaluating Centrality Indices in Real-world Networks. In *Proc. of Int. Conf. on Parallel Processing (ICPP)*.

[3] Cameron Bradley, Ghadeer Ahmed H Alabandi, and Martin Burtscher. 2025. Fringe-SGC: Counting Subgraphs with Fringe Vertices. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '25)*. Association for Computing Machinery, New York, NY, USA, 1510–1523. doi:10.1145/3712285.3759839

[4] Marco D'Antonio, Thai Son Mai, Philippas Tsigas, and Hans Vandierendonck. 2025. Wasp: Efficient Asynchronous Single-Source Shortest Path on Multicore Systems via Work Stealing. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '25)*. Association for Computing Machinery, New York, NY, USA, 2109–2125. doi:10.1145/3712285.3759872

[5] Antonio De Caro, Gennaro Cordasco, Federico Ficarelli, and Biagio Cosenza. 2025. SIGMo: High-Throughput Batched Subgraph Isomorphism on GPUs for Molecular Matching. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '25)*. Association for Computing Machinery, New York, NY, USA, 1524–1538. doi:10.1145/3712285.3759782

[6] Long Deng, Yongkun Li, Zaigui Zhang, Yinlong Xu, and John C. S. Lui. 2025. Bubble: Towards Scalable Evolving Graph Processing via Mini-Batch Sorting. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '25)*. Association for Computing Machinery, New York, NY, USA, 1553–1571. doi:10.1145/3712285.3759897

[7] Rainer Gemulla, Wolfgang Lehner, and Peter J Haas. 2008. Maintaining bounded-size sample synopses of evolving datasets. *The VLDB Journal* 17, 2 (2008), 173–201.

[8] Michael S. Gilbert, Kamesh Madduri, Erik G. Boman, and Siva Rajamanickam. 2024. Jet: Multilevel Graph Partitioning on Graphics Processing Units. *SIAM Journal on Scientific Computing* 46, 5 (2024), B700–B724. doi:10.1137/23M1559129

[9] Joseph E. Gonzalez et al. 2012. PowerGraph: Distributed graph-parallel computation on natural graphs. In *OSDI*.

[10] Joseph E. Gonzalez et al. 2014. GraphX: Graph processing in a distributed dataflow framework. In *OSDI*.

[11] Roger W Hockney. 1994. The communication challenge for MPP: Intel Paragon and Meiko CS-2. *Parallel computing* 20, 3 (1994), 389–398.

[12] Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association* 58, 301 (1963), 13–30.

[13] Madhav Jha, C Seshadhri, and Ali Pinar. 2015. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *Proceedings of the 24th international conference on world wide web*. 495–505.

[14] Madhav Jha, C Seshadhri, and Ali Pinar. 2015. A space-efficient streaming algorithm for estimating transitivity and triangle counts using the birthday paradox. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 9, 3 (2015), 1–21.

[15] Shubhashish Kar and Shaikh Arifuzzaman. 2024. MESM: A Query-Agnostic and Memory-Efficient Parallel Subgraph Matching Algorithm. In *IEEE High Performance Extreme Computing Conference, HPEC 2024*. 1–7. doi:10.1109/HPEC62836.2024.10938524

[16] George Karypis. 2011. METIS and ParMETIS. In *Encyclopedia of parallel computing*. Springer, 1117–1124.

[17] Jure Leskovec and Christos Faloutsos. 2006. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 631–636.

[18] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[19] Sebastian Lüderssen, Stefan Neumann, and Pan Peng. 2026. Four-Cycle Counting in Low-Degeneracy Graph Streams. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*. 983–994.

[20] Grzegorz Malewicz et al. 2010. Pregel: a system for large-scale graph processing. In *SIGMOD*.

[21] Andrew McGregor. 2014. Graph stream algorithms: a survey. *SIGMOD Record* (2014).

[22] Andrew McGregor and Sofya Vorotnikova. 2020. Triangle and four cycle counting in the data stream model. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 445–456.

[23] Mark Ortmann and Ulrik Brandes. 2017. Efficient orbit-aware graphlet counting. In *WWW*.

[24] Serafeim Papadias, Zoi Kaoudi, Varun Pandey, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. 2024. Counting butterflies in fully dynamic bipartite graph streams. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2917–2930.

[25] Ali Pinar, C. Seshadhri, and V. Vishal. 2017. Escape: Efficiently counting all 5-vertex subgraphs. In *WWW*.

[26] Ryan A. Rossi and Nesreen K. Ahmed. 2013. Network Repository. http://networkrepository.com

[27] Ryan A Rossi, Rong Zhou, and Nesreen K Ahmed. 2018. Estimation of graphlet counts in massive networks. *IEEE transactions on neural networks and learning systems* 30, 1 (2018), 44–57.

[28] Seyed-Vahid Sanei-Mehri, Ahmet Erdem Sariyuce, and Srikanta Tirthapura. 2018. Butterfly counting in bipartite networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2150–2159.

[29] John G Saw, Mark CK Yang, and Tse Chin Mo. 1984. Chebyshev inequality with estimated mean and variance. *The American Statistician* 38, 2 (1984), 130–132.

[30] C. Seshadhri, Ali Pinar, and Tamara G. Kolda. 2013. Wedge sampling for computing clustering coefficients and triangle counts on large graphs. In *KDD*.

[31] Lorenzo De Stefani, Alessandro Epasto, Matteo Riondato, and Eli Upfal. 2016. TRIÈST: Counting local and global triangles in fully-dynamic streams with fixed memory size. In *KDD*.

[32] Trevor Steil, Geoffrey Sanders, and Roger Pearce. 2021. Towards distributed square counting in large graphs. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 1–7.

[33] Yuxin Tang, Mangesh Bendre, and Mahashweta Das. 2024. Monarch: Distributed Butterfly Counting for Large-scale Bipartite Graph. In *2024 IEEE International Conference on Big Data (BigData)*. IEEE, 799–804.

[34] Charalampos E. Tsourakakis et al. 2009. DOULION: Counting triangles in massive graphs with a coin. In *KDD*.

[35] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. 2019. Vertex priority based butterfly counting for large-scale bipartite networks. *PVLDB* (2019).