Distributed Estimation of Graph 4-Profiles

Ethan R. Elenberg, Karthikeyan Shanmugam,
Michael Borokhovich, Alexandros G. Dimakis
The University of Texas at Austin

June 19, 2015

Abstract

We present a novel distributed algorithm for counting all four-node induced subgraphs in a big graph. These counts, called the 4-profile, describe a graph's connectivity properties and have found several uses ranging from bioinformatics to spam detection. We also study the more complicated problem of estimating the local 4-profiles centered at each vertex of the graph. The local 4-profile embeds the vertex in an 11-dimensional space that characterizes the local geometry of its neighborhood: vertices that connect different clusters will have different local 4-profiles compared to those that are only part of one dense cluster.

Our algorithm is a local message-passing scheme on the graph and computes all the local 4-profiles in parallel. We rely on two novel theoretical contributions: we show that local 4-profiles can be calculated using compressed two-hop information and also establish novel concentration results that show that graphs can be substantially sparsified and still retain good approximation quality for the global 4-profile.

We empirically evaluate our algorithm using a distributed GraphLab implementation that we scaled up to 640 cores. We show that our algorithm can compute global and local 4-profiles of graphs with millions of edges in a few minutes, significantly improving upon the previous state of the art.

1 Introduction

Graph k-profiles are local statistics that count the number of small subgraphs in a big graph. k-profiles are a natural generalization of triangle counting and are increasingly popular for several problems in big graph analytics. Globally, they form a concise graph description that has found several applications for the web [1, 2] as well as social and biological networks [3, 4]. Furthermore, as we explain, the *local* profile of a vertex is an embedding in a low-dimensional feature space that reveals local structural information. Mathematically, k-profiles are of significant recent interest since they are connected to the emerging theory of graph homomorphisms, graph limits and graphons [5, 3, 6].

There are 4 possible graphs on 3 vertices, labeled as $H_0, H_2, \ldots, H_3$, as in Figure 1 (left). The (global) 3-profile of a graph $G(V, E)$ is a vector having one coordinate for each distinct H_i that counts how many times that H_i appears as an induced subgraph of G. For example, the graph $G = K_4$ (the complete graph on 4 vertices) has the 3-profile $[0, 0, 0, 4]$ since it contains 4 triangles and no other (induced) subgraphs. The graph C_5 (the cycle on 5 vertices, *i.e.* a pentagon) has the 3-profile $[0, 5, 5, 0]$. Note that the sum of the k-profile is always $\binom{|V|}{k}$, the total number of subgraphs. Estimating 3-profiles of big graphs is a topic that has received attention from several communities recently (e.g. see [3, 7, 8, 9] and references therein).

In this paper we are interested in the significantly more challenging problem of estimating 4-profiles. Figure 1 (right), shows the 11 possible graphs on 4 vertices, labeled as $F_i,\ i = 0 \ldots 10$. Given a big graph $G(V, E)$ we are interested in estimating the global 4-profile, i.e. count how many times each F_i appears as an induced subgraph of G. In addition to global graph statistics, we are interested in local 4-profiles: Given a specific vertex v_0, the local 4-profile of v_0 is an 11-dimensional vector[1] with each coordinate i counting, how many induced F_i's contain v_0. In Figure 2 we show an example of the local 4-profile of a vertex.

[1] Actually there are 17 local subgraphs when considering vertex automorphisms. This is discussed in Section 3 and in the Appendix in detail. For the purpose of initial exposition, we will ignore vertex automorphisms.

The local 4-profile of a vertex can be seen as an embedding in an 11-dimensional space that characterizes the local geometry of its neighborhood: vertices that connect different clusters will have different local 4-profiles compared to those that are only part of one dense cluster. A very naive estimation of 4-profiles requires examining $\binom{n}{4}$ possible subgraphs. Furthermore, for estimating each local 4-profile independently, this computation has to be repeated n times, once for each vertex. Note that local 4-profiles contain more information compared to the global profile: they can be added and appropriately rescaled (using the symmetries of each subgraph, also called automorphism orbits [4]) to obtain the global 4-profile. Since some of the 4-profile subgraphs are disconnected (like F_0, F_1, F_5), local 4-profiles contain information beyond the local neighborhood of a vertex. Therefore in a distributed setting it seems that global communication is required.

1.1 Our contributions

Surprisingly, we show that very limited global information is sufficient to calculate all local 4-profiles and that it can be re-used to calculate all the local 4-profiles in parallel. Specifically, we introduce a distributed algorithm to estimate all the local 4-profiles and the global profile of a big graph. Our algorithm relies on two novel theoretical results:

Two-hop histograms are sufficient: Our algorithm operates by having each vertex perform local message-passing to its neighbors. Focusing on a vertex v_0, the first easy step is to calculate its local 3-profile. It can be seen that the local 3-profile combined with the full two-hop neighborhood information is sufficient to estimate the local 4-profile for each vertex v_0. However, we show that less information needs to be communicated. Specifically, we prove that what we call *two-hop histogram* is sufficient: for each vertex v_i that is 2-hops from v_0, we only need the *number of distinct paths* connecting it to v_0, not the full two hop neighborhood. If the two-hop neighborhood is a tree, this amounts to no compression. However, for real graphs the two-hop histogram saves a factor of 3x to 5x in communication in our experiments. This enables (Section 4) an even more significant running-time speedup of $5-10$ times on several distributed experiments using $12-20$ compute nodes.

Profile Sparsification: One idea that originated from triangle counting [10] is to first perform random subsampling of edges to create a sparse graph called a *triangle sparsifier*. Then count the number of triangles in the sparse graph and re-scale appropriately to estimate the number in the original graph. The main challenge is proving that the randomly sparsified graph has a number of triangles sufficiently concentrated around its expectation. Elenberg et al. [9] recently generalized this idea for 3-profile sparsifiers and showed concentration results for estimating the full 3-profile. These papers rely on Kim-Vu polynomial concentration techniques [11] that scale well in theory but, typically the estimated errors are orders of magnitude larger than the measured quantities for reasonable graph sizes. In this paper, we introduce novel concentration bounds for k-profile sparsifiers that use a novel information theoretic technique called read-k functions [12]. Our read-k bounds allow usable concentration inequalities for sparsification factors of approximately 0.4 or higher (Section 4.1). Note that removing half the edges of the graph does not accelerate the running time by a factor of 2, but rather by a factor of nearly 8, as shown in our experiments.

System implementation and evaluation: We implemented our algorithm using GraphLab [13] and tested in multicore and distributed systems scaling up to 640 cores. The benefits of two-hop histogram compression and sparsification allowed us to compute the global and local 4-profiles of very large graphs. For example, for a graph with with 5 million vertices and 40 million edges we estimated the global 4-profile in less than 10 seconds. For computing all local 4-profiles on this graph the previous state of the art [8] required 1200 seconds while our algorithm required less than 100 seconds.

1.2 Related work

The problem of counting triangles in a graph has been addressed in distributed settings [14], and this is a standard analytics task for graph engines [15]. The Doullion algorithm [16] estimates a graph's triangle count via simple edge subsampling. Other recent work analyzes more complex sampling schemes [17] and extends to approximately counting certain 4-subgraphs [18, 19]. Mapreduce algorithms for clique counting were introduced by Finocchi et al. [20]. Our approach is most closely related to that of [9], which calculates all 3-subgraphs and a subsest of 4-subgraphs distributedly.

The breakthrough work of Kim and Vu [11] uses multivariate polynomials to provide asymptotic bounds

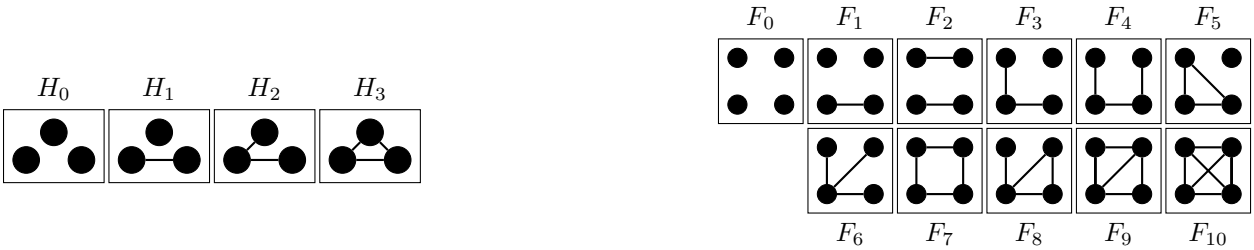


Figure 1: Left: The 4 possible non-isomorphic graphs on 3 vertices used to calculate the 3-profile of a graph G. The 3-profile counts how many times each of H_i appears in G. Right: The 11 non-isomorphic graphs on 4 vertices used to calculate the 4-profile of a graph.

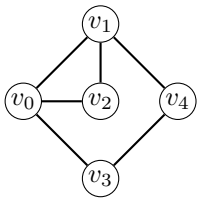


Figure 2: An example for local profiles. The global 3-profile is $(0, 3, 6, 1)$. The global 4-profile is $(0, 0, 0, 0, 2, 0, 0, 1, 2, 0, 0)$. The local 4-profile of v_0 is $(0, 0, 0, 0, 1, 0, 0, 1, 2, 0, 0)$. The first 1 in the profile corresponds to the subgraph F_4. Notice that v_0 participates in only one F_4, jointly with vertices v_2, v_3, v_4.

on the number of triangles in a random graph much better than McDiarmid's inequality or simple Chernoff bounds. This was later improved and generalized in [21], and applied to subsampled triangle counting in [10]. Our analysis uses read-k functions [12] to produce sharper concentration results in practical situations.

Previous systems of equations relating clique counts to other 4-subgraphs appear in [22], [7], and [8]. However, these are applied in a centralized setting and depend on information from nonadjacent vertices. In this work, we use additional equations to solve the same system by sharing only local information. The connected 4-subgraphs, or graphlets [4], have found applications in fields such as bioinformatics [23] and computational neuroscience [24]. In [25], authors use *all* global 4-subgraphs to analyze neuronal networks. We evaluate our algorithm against Orca [8], a centralized 4-graphlet counting algorithm, as well as its GPU implementation [26]. Notice that while Orca calculates only connected 4-subgraphs, our algorithm calculates all the connected and the disconnected 4-subgraphs for each vertex.

2 Sparsification and Concentration

In this section, we describe the process for estimating the exact number of subgraphs in a graph G. Denote the exact counts by $[N_0 \dots N_{10}]^T$ and the estimates by $[X_0 \dots X_{10}]^T$.

We are sparsifying the orignal graph G by keeping each edge independently with probability p. Denote the random sub-sampled graph with $\tilde{G}$ and its global 4-profile by $[Y_0 \dots Y_{10}]^T$. Clearly each triangle survives with probability p^3 and each 4-clique survives with p^6. Therefore in expectation, $\mathbb{E}[Y_{10}] = \frac{1}{p^6} N_{10}$.

This simple correspondence does not hold for other subgraphs: each triangle in $\tilde{G}$ can only be a triangle in G that survived edge removals, but other subgraphs of $\tilde{G}$ could be originating from multiple subgraphs of G depending on the random sparsification process. We can, however, relate the original 4-profile vector to the expected subsampled 4-profile vector by a matrix multiplication. Let $F(abcd)$ and $\tilde{F}(abcd)$ represent the induced 4-subgaph on the vertices $abcd$ before and after subsampling, respectively. Then define $\mathbf{H}$ by $H_{ij} = \mathbb{P}(\tilde{F}(abcd) = F_i \,|\, F(abcd) = F_j)$. Thus, we form an unbiased estimator, *i.e.* $\mathbb{E}[X_i] = N_i,\ i = 1, \dots, 10$, by inverting the edge sampling matrix.

For 3-profiles, this process is described by the following system of equations:

$$\begin{bmatrix} \mathbb{E}[Y_0] \\ \mathbb{E}[Y_1] \\ \mathbb{E}[Y_2] \\ \mathbb{E}[Y_3] \end{bmatrix} = \begin{bmatrix} 1 & 1-p & (1-p)^2 & (1-p)^3 \\ 0 & p & 2p(1-p) & 3p(1-p)^2 \\ 0 & 0 & p^2 & 3p^2(1-p) \\ 0 & 0 & 0 & p^3 \end{bmatrix} \begin{bmatrix} n_0 \\ n_1 \\ n_2 \\ n_3, \end{bmatrix} \tag{1}$$

For 4-profiles, the vectors are 11 dimensional and a similar linear system can be explicitly computed- we include the equations in the appendix. This matrix turns out to be invertible and we can therefore calculate the 4-profile exactly if we have access to the expected values of the sparsified 4-profile. Of course, we can only obtain one sample random graph and calculate that 4-profile, which will be an accurate estimate if the 4-profile quantities are sufficiently concentrated around their expectation.

Graph k-profile concentration: Previous work used this idea of graph sparsification for triangle counting [10] and 3-profiles [9]. The main concentation tool used was the Kim and Vu polynomial concentration [11, 10] which unfortunately gives very loose concentration bounds for practical graph sizes.

In this paper we establish novel concentration results that exploit the fact that partial derivatives of the desired quantities are sparse in the number of edge variables. This allows us to use a novel information theoretic concentration technique called read-k functions [12].

For simplicity, we only explain the concentration of 4-cliques (F_{10} subgraphs) here. We establish the general result for all 11 4-profile variables in the appendix. Our main concentration result is as follows:

Theorem 1. *Let G be a graph with N_{10} 4-cliques, and let k_{10} be the maximum number of 4-cliques sharing a common edge. Let X_{10} be the 4-clique estimate obtained from subsampling each edge with probability $0 < p \leq 1$, choose $0 < \delta < 1$, and choose $\epsilon_{RK} > 0$. If*

$$p \geq \left(\frac{\log(2/\delta) k_{10}}{2\epsilon_{RK}^2 N_{10}} \right)^{1/12},$$

then $|N_{10} - X_{10}| \leq \epsilon_{RK} N_{10}$ with probability at least $1 - \delta$.

Proof. Our proof relies on read-k function families [12], a recent characterization of dependencies among functions of random variables. See the Appendix for full details. □

Next, we state conditions under which our method outperforms the Kim and Vu concentration results [11]. Proof can be found in the Appendix:

Corollary 1. *Let G be a graph with m edges. If $p = \Omega(1/\log m)$ and $\delta = \Omega(1/m)$, then read-k provides better triangle sparsifier accuracy than Kim-Vu. If additionally $k_{10} \leq N_{10}^{5/6}$, then read-k provides better 4-clique sparsifier accuracy than Kim-Vu.*

We note that the asymptotic condition above includes a very small constant term. This implies our concentration result is superior over all p values of practical interest. This is also investigated in Section 4.1 for some realistic graphs.

3 Distributed Algorithm

In this section, we describe 4-Prof-Dist, our algorithm for computing the exact 4-profiles in a distributed manner. To the best of our knowledge, this is the first distributed algorithm for calculating 4-profiles. The key insight is to cast existing and novel equations into the GraphLab PowerGraph framework [13] to get implicit connectivity information about vertices outside the 1-hop neighborhood. Specifically, we construct the local 4-profile from local 3-profile, local 4-clique count, and additional histogram information which describes the number of paths to all 2-hop neighbors.

Theorem 2. *There is a distributed algorithm that computes the local 4-profile of a graph given each vertex has stored its local 3-profile, triangle list, and 2-hop histogram.*

4-Prof-Dist is implemented in the Gather-Apply-Scatter (GAS) framework [13]. A distributed algorithm in this framework has 3 main phases: Gather, Apply and Scatter. Every vertex and edge has stored data which is acted upon. During the Gather phase, a vertex can access all its adjacent edges and neighbors and *gather* data they possess, *e.g.*, neighbor id, using a custom reduce operation $\oplus$ (*e.g.*, addition, concatenation). The accumulated information is available for a vertex at the next phase, Apply, in which it can change its own data. In the final Scatter phase, every edge sees the data of its (incident) vertices and operates on it to modify the edge data. All nodes start each phase simultaneously, and if needed, the whole GAS cycle is repeated until algorithm's completion.

4-Prof-Dist solves a slightly larger problem of keeping track of counts of 17 unique subgraphs up to vertex automorphism (see Appendix, Figure 6). We will describe a full rank system of equations which is sufficient to calculate the local 4-profile at every $v \in V$. The following subsections each explain a component of 4-Prof-Dist. These separate routines are combined efficiently in Algorithm 1 to calculate the local 4-profile in a small number of GAS cycles.

3.1 Edge pivot equations

The majority of our equations relate the local 4-profile to neighboring local 3-profiles with edge pivots. Counting specific pairs of 3-subgraphs formed with an edge va and *gathering* over all neighbors a computes a linear combination of the local subgraph counts at v. Before that, in an initial GAS round, each vertex v must gather the id of each vertex in its neighborhood, *i.e.* $a \in \Gamma(v)$, and the following quantities must be stored at each edge va during Scatter phase[2]:

$$n_{3,va} = |\Gamma(v) \cap \Gamma(a)|,\ n^e_{1,va} = |\overline{\Gamma(v) \ominus \Gamma(a)}|,\ n^c_{2,va} = |\Gamma(a) \backslash \{\Gamma(v) \cup v\}|,\ n^e_{2,va} = n^c_{2,av}. \tag{2}$$

Gather: The above quantities can be summed at each vertex v to calculate the local 3-profile at v. For example, $n_{3,v} = \frac{1}{2}\sum_a n_{3,va}$. In addition, we gather the sum of products (and linear combinations) of some pairs of these quantities forming what we call 13 *edge pivot* equations. For example:

$$\sum_{a \in \Gamma(v)} \binom{n_{3,va}}{2} = F_9^{'}(v) + 3F_{10}(v), \qquad \sum_{a \in \Gamma(v)} n^c_{2,va} n^e_{2,va} = F_4^{'}(v) + 2F_7(v). \tag{3}$$

The primed notation differentiates between subgraphs of different automorphism orbits. See the appendix for full all edge-pivot equations. By accumulating pairs of 3-profile structures as in (3), we receive aggregate connectivity information about vertices more than 1 hop away. In the second equation, for example, in the 4 node graphs formed by the product between $n_{2,va}$ and $n^e_{2,va}$ subgraphs on edge va, two additional vertices may be either disjoint or connected. Thus, $F_4^{'}(v)$ and $F_7(v)$ both contribute to the sum of $n^c_{2,va} n^e_{2,va}$.
Apply: The left hand sides of all such equations are stored at v.

3.2 Clique counting

The aim of this subtask is to count 4 cliques that contain v. For this, we accumulate a list of triangles at each vertex v and then at the scatter stage at every va, it is possible check if common neighbors have an edge between them implying a 4 clique.
Scatter: In addition to the size of $|\Gamma(v) \cap \Gamma(a)|$ at each edge va as before, we now require the intersection list $\{b : b \in \Gamma(v), b \in \Gamma(a)\}$ as a starting point.
Gather, Apply: The intersection list, denoted $\Delta(v)$, is gathered at each vertex v. This produces all pairs of neighbors in $\Gamma(v)$ which are adjacent, *i.e.* all triangles containing v. $\Delta(v)$ is stored at v during Apply.
Gather, Apply: Each edge va computes the number of 4 cliques by counting how many pairs in $\Delta(a)$ contain exactly two neighbors of v:

$$\sum_{a \in \Gamma(v)} |(b, c) \in \Delta(a) : b \in \Gamma(v), c \in \Gamma(v)| = 3F_{10}(v). \tag{4}$$

We use triangle list at each vertex to calculate $F_8(v)$ concurrently. At the Apply stage, store the left hand sides as vertex data.

[2]The notation $A \ominus B$ represents symmetric difference of sets A and B

3.3 Histogram 2-hop information

Instead of calculating the number of cycles $F_7(v)$ directly, we can simply construct another linearly independent equation and add it to our system. Let each vertex a have a vector of (vertex id, count) pairs $(p, c_a[p])$ for each of its adjacent vertices p. A neighboring vertex v can collect this information to determine how many paths exist from v to p through any $a \in \Gamma(v)$. This lets us calculate a combination of cycle subgraphs with an equation that is linearly independent from the others in our system. The reader is referred to the Appendix for a more thorough analysis of the communication savings and storage requirements for this data structure.

Gather: At each v, we take a union of histograms from each neighbor a, resolving duplicate entries with the reduce operation $(p, a_1) \oplus (p, a_2) = (p, a_1 + a_2)$.

Apply: Given the gathered histogram vector $\{\oplus_{a \in \Gamma(v)} c_a[p]\}_{p \notin \Gamma(v)}$, calculate the number of intersecting 2-paths to p through a pair of neighbors:

$$\sum_{p \notin \Gamma(v)} \binom{\oplus_{a \in \Gamma(v)} c_a[p]}{2} = F_7(v) + F_9(v). \tag{5}$$

Algorithm 1 4-Prof-Dist

1: **Input:** Graph $G(V, E)$ with $|V|$ vertices, $|E|$ edges
2: **Gather:** For each vertex v union over edges of the 'other' vertex in the edge, $\cup_{a \in \Gamma(v)} a = \Gamma(v)$
3: **Apply:** Store the gather as vertex data `v.nb`, size automatically stored
4: **Scatter:** For each edge e_{va}, compute and store scalars in (2).
5: **Gather:** For each edge e_{va}, sum edge scalar data of neighbors in (3) and combine histograms.
6: **Apply:** For each vertex v, sum over $p \notin \Gamma(v)$ in (5), store other data in array `v.u`. No Scatter.
7: **Gather:** For each vertex v collect pairs of connected neighbors in $\Delta(v)$.
8: **Apply:** Store connected neighbor (triangle) list as vertex data `v.conn`. No Scatter.
9: **Gather:** For each vertex v sum (4).
10: **Apply:** Append data to array `v.u`. Multiply `v.u` by a matrix to solve system of equations.
11: **return** [`v: v.N0 v.N1 v.N2 ... v.N10`]

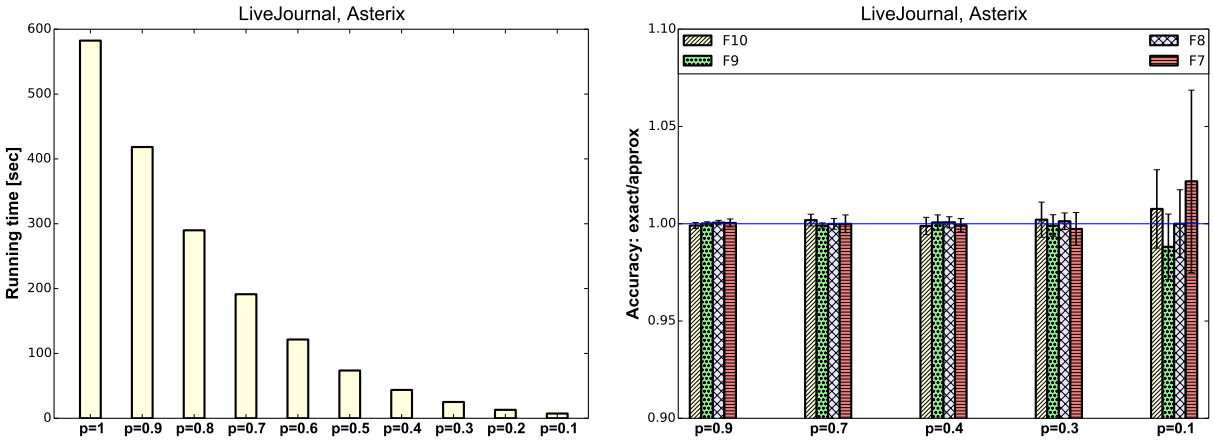
4 Experiments

Let us now describe the implementation and experimental results of our algorithm. We implement 4-Prof-Dist on GraphLab v2.2 (PowerGraph) [13] and measure its running time and accuracy on large input graphs. First, we show that edge sampling yields very good approximation results for global 4-profiles counts and achieves substantial execution speedup and saving in network traffic when multiple machines are in use. Due to its distributed nature, we can show 4-Prof-Dist runs substantially faster when using multiple CPU cores and/or machines. Notice that multicore and multiple machines can not speed up some centralized algorithms, e.g., Orca [8], which we use as a baseline for our results. However, Orca produces only a partial 4-subgraph count, i.e., it calculates only connected 4-subgraphs, while 4-Prof-Dist calculates all the 17 per vertex.

The systems: We perform the experiments on two systems. The first system is a single powerful server, further referred to as Asterix. The server is equipped with 256 GB of RAM and two Intel Xeon E5-2699 v3 CPUs, 18 cores each. Since each core has two hardware threads, up to 72 logical cores are available to the GraphLab engine. The second system is an EC2 cluster on AWS[3]. The cluster is comprised of 20 c3.8xlarge machines, each having 60 GB RAM and 32 virtual CPUs.

The data: In our experiments we use two real graphs representing different datasets: social networks (LiveJournal: 4,846,609 vertices, 42,851,237 edges) and a WWW graph of Notre Dame (WEB-NOTRE: 325,729 vertices, 1,090,108 edges). Notice that the above graphs are originally directed, but since our work deals with undirected graphs, all duplicate edges (i.e., bi-directional) were removed and directionality is ignored.

[3]Amazon Web Services - http://aws.amazon.com



(a) (b)

Figure 3: LiveJournal graph, Asterix system. All the results are averaged over 10 iterations. (a) – Running time as a function of sampling probability. (b) – Accuracy of the $F_7 - F_{10}$ global counts. Measured as ratio of the exact count and the estimated.

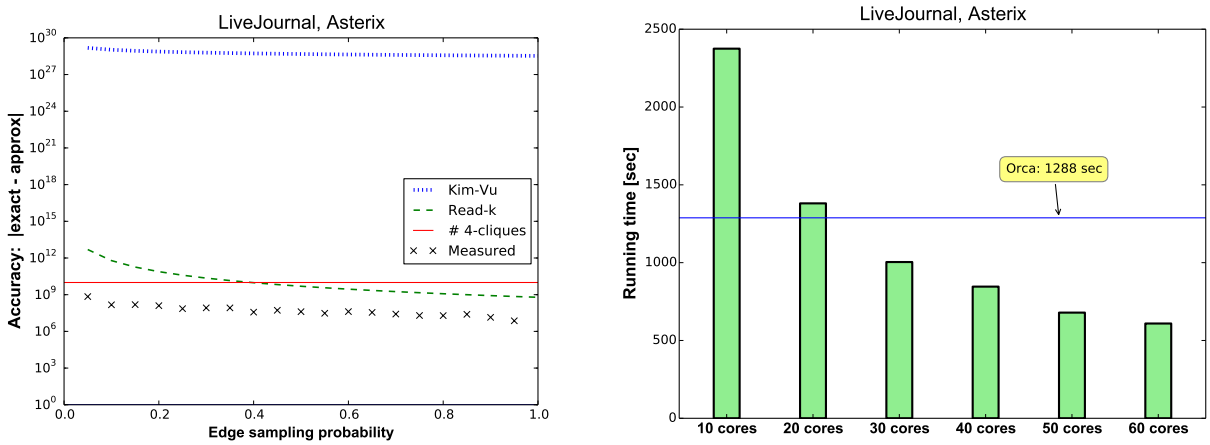
4.1 Results

Accuracy: The first result is that our edge sampling approach greatly improves running time while maintaining a very good approximate of the *global* 4-profile. In Figure 3a we can see that the running time decreases drastically when the sampling probability decreases. At the same time, Figure 3b shows that the mean ratio of true to estimated global 4-profiles is within $\pm 2.5\%$. We show here only profiles $F_7 - F_{10}$ since their counts are the smallest and were observed to have the lowest accuracy. In Figure 4a we compare theoretical concentration bounds on a logarithmic scale and show the benefit of Theorem 1. While the guarantees provided by Kim-Vu [11] bounds are very loose (the additive error is bounded by numbers which are orders of magnitude larger than the true value), the read-k approach is much closer to the measured values. We can see that even for small sampling probability, the measured error is 1-2 orders of magnitude smaller than the value prediced by Theorem 1.

2-hop histogram: Now we compare two methods of calculating the left hand side of (5) from Section 3.3. We show that a simple implementation in which a vertex gathers its full 2-hop neighborhood (i.e., IDs of its neighbors' neighbors) is much less efficient than the *2-hop histogram* approach used in 4-Prof-Dist (see Section 3.3). In Figures 5a and 5c we can see that the *histogram* approach is an order of magnitude faster for various numbers of machines, and that its network requirements are substantially less than of the simple implementation. Moreover, our algorithm could handle much larger graphs while the the simple implementation ran out of memory.

Running time: Finally, we show that 4-Prof-Dist can run much faster current state-of-the-art graphlet counting implementations. The algorithm and the GraphLab platform on which it runs are both distributed in nature. The latter allows 4-Prof-Dist to exploit multiple cores on a single machine as well as a cluster of machines. Figure 4b shows running time as a function of CPU cores. We compare this result to the running time of a single core, C++ implementation of Orca [8]. Our 4-Prof-Dist algorhm becomes faster after only 25 cores and is 2x faster using 60 cores. Moreover, 4-Prof-Dist allows scaling to a large number of machines. In Figure 5b we can see how the running time for the LiveJournal graph decreases when the number of machines increases. Since Orca cannot benefit from multiple machines, we see that 4-Prof-Dist runs up to 12x faster than Orca. This gap widens as the cluster grows larger. In [26], the authors implemented a GPU version of Orca using CUDA. However, the reported speedup is about 2x which is much less than we show here on the AWS cluster (see Figure 5b for $p = 1$). We also note a substantial running time benefit of the sampling approach for *global* 4-profiles. In Figures 5b and 5d we see that with $p = 0.1$ we can achieve order of magnitude improvements in both speed and network traffic, while maintaining very good accuracy (see Figure 3b).

Conclusions: In short, we have used read-k function families to provide meaningful concentration results



(a) (b)

Figure 4: LiveJournal graph, Asterix system. All the results are averaged over 10 iterations. (a) – Comparison of 4-clique sparsifier concentration bounds with accuracy measured in edge sampling experiments. (b) – Comparison of running times of Orca and our 4-Prof-Dist algorithms. Clearly, 4-Prof-Dist benefits from he multitude of cores.

for the 4-clique and 4-profile sparsifier problems. Our novel distributed algorithm is faster than a state of the art graphlet counting algorithm by an order of magnitude. An important open problem we plan to investigate in the future is the usage of local 4-profile vectors as features for graph vertices for various learning tasks.

References

[1] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis, "Efficient semi-streaming algorithms for local triangle counting in massive graphs," in *KDD*, 2008.

[2] D. O'Callaghan, M. Harrigan, J. Carthy, and P. Cunningham, "Identifying Discriminating Network Motifs in YouTube Spam," Feb. 2012.

[3] J. Ugander, L. Backstrom, M. Park, and J. Kleinberg, "Subgraph Frequencies: Mapping the Empirical and Extremal Geography of Large Graph Collections," in *WWW*, 2013.

[4] N. Przulj, "Biological Network Comparison Using Graphlet Degree Distribution," *Bioinformatics*, vol. 23, no. 2, pp. 177–183, 2007.

[5] C. Borgs, J. Chayes, and K. Vesztergombi, "Counting Graph Homomorphisms," *Topics in Discrete Mathematics*, pp. 315–371, 2006.

[6] L. Lovász, *Large networks and graph limits*. American Mathematical Soc., 2012, vol. 60.

[7] V. V. Williams, J. Wang, R. Williams, and H. Yu, "Finding Four-Node Subgraphs in Triangle Time," *SODA*, pp. 1671–1680, 2014. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/1.9781611973730.111

[8] T. Hočevar and J. Demšar, "A combinatorial approach to graphlet counting." *Bioinformatics*, vol. 30, no. 4, pp. 559–65, Feb. 2014. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/24336411

[9] E. R. Elenberg, K. Shanmugam, M. Borokhovich, and A. G. Dimakis, "Beyond Triangles: A Distributed Framework for Estimating 3-profiles of Large Graphs," in *KDD*, 2015.

[10] C. E. Tsourakakis, M. Kolountzakis, and G. L. Miller, "Triangle Sparsifiers," *Journal of Graph Theory and Applications*, vol. 15, no. 6, pp. 703–726, 2011.

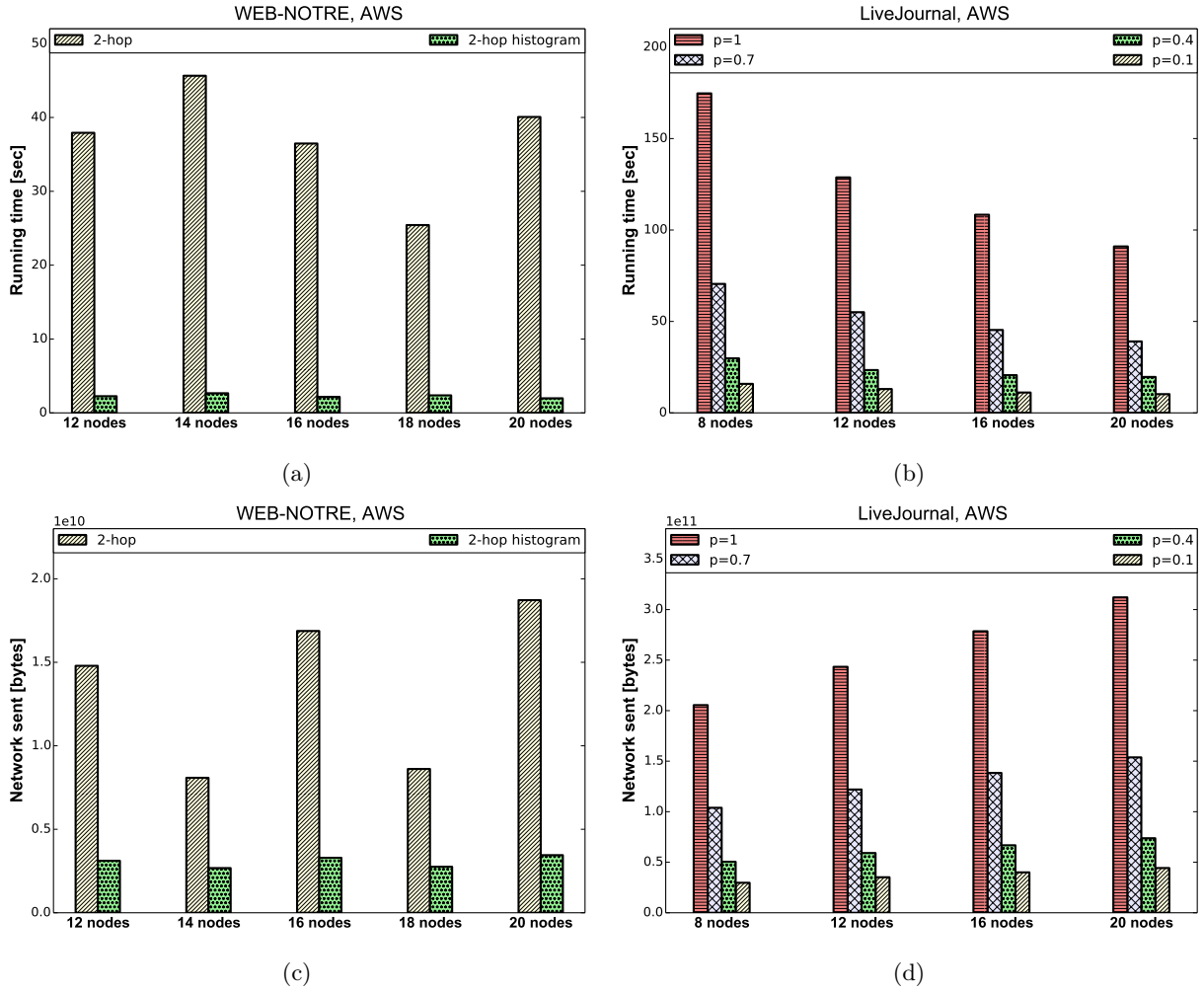


Figure 5: AWS cluster of up to 20 machines (nodes). (a,c) – Running time and network usage comparing naive 2-hop implementation and 2-hop histogram approach on the Notre Dame web graph. (b,d) – Running time and network usage of 4-Prof-Dist for various number of compute nodes and sampling probability p, on the LiveJournal graph. All results are averages over 10 iterations.

[11] J. H. Kim and V. H. Vu, "Concentration of Multivariate Polynomials and Its Applications," *Combinatorica*, vol. 20, no. 3, pp. 417–434, 2000.

[12] D. Gavinsky, S. Lovett, M. Saks, and S. Srinivasan, "A Tail Bound for Read-k Families of Functions," pp. 1–8, 2012. [Online]. Available: http://arxiv.org/pdf/1205.1478v1.pdf

[13] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs," in *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2012, pp. 17–30.

[14] T. Schank, "Algorithmic Aspects of Triangle-Based Network Analysis," Ph.D. dissertation, 2007.

[15] N. Satish, N. Sundaram, M. A. Patwary, J. Seo, J. Park, M. A. Hassaan, S. Sengupta, Z. Yin, and P. Dubey, "Navigating the Maze of Graph Analytics Frameworks using Massive Graph Datasets," in *SIGMOD*, 2014, pp. 979–990.

[16] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos, "DOULION: Counting Triangles in Massive Graphs with a Coin," in *SIGKDD*, 2009.

[17] C. Seshadhri, A. Pinar, and T. G. Kolda, "Triadic measures on graphs: The power of wedge sampling," in *Proceedings of the SIAM Conference on Data Mining*, 2013, pp. 10–18.

[18] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella, "Graph Sample and Hold: A Framework for Big-Graph Analytics," in *KDD*, 2014.

[19] M. Jha, C. Seshadhri, and A. Pinar, "Path Sampling: A Fast and Provable Method for Estimating 4-Vertex Subgraph Counts," 2014.

[20] I. Finocchi, M. Finocchi, and E. G. Fusco, "Clique counting in MapReduce: theory and experiments," Mar. 2014. [Online]. Available: http://arxiv.org/abs/1403.0734

[21] S. Janson, K. Oleszkiewicz, and A. Ruciński, "Upper tails for subgraph counts in random graphs," *Israel Journal of Mathematics*, vol. 142, no. 1, pp. 61–92, 2004. [Online]. Available: http://dx.doi.org/10.1007/BF02771528

[22] M. Kowaluk, A. Lingas, and E.-M. Lundell, "Counting and detecting small subgraphs via equations," *SIAM Journal of Discrete Mathematics*, vol. 27, no. 2, pp. 892–909, 2013.

[23] N. Shervashidze, K. Mehlhorn, and T. H. Petri, "Efficient graphlet kernels for large graph comparison," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2009, pp. 488–495.

[24] F. Fei, B. Jie, and D. Zhang, "Frequent and Discriminative Subnetwork Mining for Mild Cognitive Impairment Classification," *Brain Connectivity*, vol. 4, no. 5, pp. 347–360, Jun. 2014. [Online]. Available: http://online.liebertpub.com/doi/abs/10.1089/brain.2013.0214

[25] V. LR, C. BL, P. E, H. DH, and C. DB, "Structural Properties of the Caenorhabditis elegans Neuronal Network," in *PLoS Comput Biol*, vol. 7(2): e1001066, 2011.

[26] A. Milinković, S. Milinković, and L. Lazicć, "A contribution to acceleration of graphlet counting," in *Infoteh Jahorina Symposium*, vol. 14, no. March, 2015, pp. 741–745.

[27] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," in *STOC*, 1996, pp. 20–29.

A Appendix

A.1 Proof of Theorem 1

Rather than Lipschitz bounding the value of each partial derivative, as in [11, 10, 9], our main technical tool [12] benefits from the fact that each first partial derivative is sparse in the number of edge variables:

Definition 1 (read-k families)**.** *Let $X_1, \ldots, X_m$ be independent random variables. For $j \in [r]$, let $P_j \subseteq [m]$ and let f_j be a Boolean function of $\{X_i\}_{i \in P_j}$. Assume that $|\{j | i \in P_j\}| \leq k$ for every $i \in [m]$. Then the random variables $Z_j = f_j(\{X_i\}_{i \in P_j})$ are called a read-k family.*

Each variable only affects k of the r Boolean functions. This is exactly the number of subgraph structures which share a common edge. We now state the main result required for our analysis.

Proposition 1 (concentration of read-k sums [12])**.** *Let $Z_1, \ldots, Z_r$ be a family of read-k indicator variables with $\mathbb{P}(Z_i = 1) = p_i$, and let p be the average of $p_1, \ldots, p_r$. Then for any $\epsilon > 0$,*

$$\mathbb{P}\left(\sum_{i=1}^{r} Z_i \geq (p+\epsilon)r\right) \leq \exp\left(-D(p+\epsilon \parallel p)\frac{r}{k}\right) \leq \exp\left(-2\epsilon^2 \frac{r}{k}\right)$$
$$\mathbb{P}\left(\sum_{i=1}^{r} Z_i \leq (p-\epsilon)r\right) \leq \exp\left(-D(p-\epsilon \parallel p)\frac{r}{k}\right) \leq \exp\left(-2\epsilon^2 \frac{r}{k}\right),$$

where $D(x \parallel y) = x \log\left(\frac{x}{y}\right) + (1-x)\log\left(\frac{1-x}{1-y}\right)$ is the Kullback-Leibler divergence of x and y.

Note that when applied to subgraph estimation, the bound in this proposition is a function of k_i and the subgraph count N_i independent of the number of edges $|E|$. Therefore, it is much stronger than arguments involving Lipschitz bounded functions such as McDiarmid's inequality. Let G be a graph with N_{10} 4-cliques and a maximum of k_{10} 4-cliques sharing a common edge. Let $Y_{10} = \sum_{\boxtimes(a,b,c,d) \in \mathcal{H}_{10}} t_{ab}t_{bc}t_{cd}t_{da}t_{ac}t_{bd}$. Then

$$\mathbb{P}\left(|Y_{10} - p^6 N_{10}| \geq \epsilon_{RK} N_{10}\right) \leq 2\exp\left(-\frac{2\epsilon^2 N_{10}}{k_{10}}\right)$$
$$\Rightarrow \mathbb{P}\left(|X_{10} - N_{10}| \geq \epsilon_{RK} N_{10}\right) = \mathbb{P}\left(|Y_{10} - p^6 N_{10}| \geq p^6 \epsilon_{RK} N_{10}\right) \leq 2\exp\left(-\frac{2p^{12}\epsilon_{RK}^2 N_{10}}{k_{10}}\right).$$

The claim follows by setting the right hand side less than δ and solving for p.

A.2 Proof of Corollary 1

We prove this result for the case of 4-cliques only because the case for triangles is similar. First we must derive a similar 4-clique concentration bound using the techniques in [11, 10].

Lemma 1. *Let G be a graph with m edges and N_{10} cliques, and k_{10} be the maximum number of 4-cliques sharing a common edge. Let $a_6 = 8^6\sqrt{6!}$, $0 < p \leq 1$, and $\epsilon_{KV} > 0$. Let X_{10} be the 4-clique estimate obtained from subsampling each edge with probability p. If*

$$\frac{p}{\max\left\{\sqrt[6]{1/N_{10}}, \sqrt[3]{k_{10}/N_{10}}\right\}} \geq \frac{a_6^2 \log^{12}(m^{5+\gamma})}{\epsilon_{KV}^2}, \tag{6}$$

then $|N_{10} - X_{10}| \leq \epsilon_{KV} N_{10}$ with probability at least $1 - \frac{1}{m^\gamma}$.

Proof. This proof is a straightforward application of the main result in [11], repeated below for completeness. Let $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m) \in \mathbb{Z}_+^m$ and define $\mathbb{E}_{\geq 1}[X] = \max_{\boldsymbol{\alpha}: \|\boldsymbol{\alpha}\|_1 \geq 1} \mathbb{E}(\partial^{\boldsymbol{\alpha}} X)$, where

$$\mathbb{E}(\partial^{\boldsymbol{\alpha}} X) = \mathbb{E}\left[(\frac{\partial}{\partial t_1})^{\alpha_1} \ldots (\frac{\partial}{\partial t_m})^{\alpha_m}\left[X(t_1, \ldots, t_m)\right]\right] \tag{7}$$

Further, we call a polynomial totally positive if the coefficients of all the monomials involved are non-negative.

Proposition 2 (Kim-Vu concentration of multivariate polynomials [11])**.** *Let Y be a random totally positive Boolean polynomial in m Boolean random variables with degree at most k. If $\mathbb{E}[Y] \geq \mathbb{E}_{\geq 1}[Y]$, then*

$$\mathbb{P}\left(|Y - \mathbb{E}[Y]| > a_k \sqrt{\mathbb{E}[Y]\mathbb{E}_{\geq 1}[Y]}\lambda^k\right) = \mathcal{O}\left(\exp\left(-\lambda + (k-1)\log m\right)\right) \tag{8}$$

for any $\lambda > 1$, where $a_k = 8^k k!^{1/2}$.

Let $Y_{10} = \sum_{\boxtimes(a,b,c,d)\in\mathcal{H}_{10}} t_{ab}t_{bc}t_{cd}t_{da}t_{ac}t_{bd}$. Clearly Y_{10} is totally positive. Let $k_{10,ab}$, σ_{abc}, and ν_{abc} be the maximum number of 4-cliques sharing a common edge t_{ab}, wedge Λ_{abc}, and triangle Δ_{abc}, respectively. Taking repeated partial derivatives,

$$\begin{aligned}
\mathbb{E}\left[\frac{\partial Y_{10}}{\partial t_{ab}}\right] &= p^5 k_{10,ab}, \\
\mathbb{E}\left[\frac{\partial Y_{10}}{\partial t_{ab}t_{bc}}\right] &= p^4\sigma_{abc}, \quad \mathbb{E}\left[\frac{\partial Y_{10}}{\partial t_{ab}t_{cd}}\right] = p^4, \\
\mathbb{E}\left[\frac{\partial Y_{10}}{\partial t_{ab}t_{bc}t_{ac}}\right] &= p^3\nu_{abc}, \quad \mathbb{E}\left[\frac{\partial Y_{10}}{\partial t_{ab}t_{bc}t_{cd}}\right] = p^3, \\
\mathbb{E}\left[\frac{\partial Y_{10}}{\partial t_{ab}t_{bc}t_{ac}t_{da}}\right] &= \mathbb{E}\left[\frac{\partial Y_{10}}{\partial t_{ab}t_{bc}t_{cd}t_{da}}\right] = p^2, \\
\mathbb{E}\left[\frac{\partial Y_{10}}{\partial t_{ab}t_{bc}t_{cd}t_{da}t_{ac}}\right] &= p, \quad \mathbb{E}\left[\frac{\partial Y_{10}}{\partial t_{ab}t_{bc}t_{cd}t_{da}t_{ac}t_{bd}}\right] = 1.
\end{aligned}$$

Noting that $\sigma_{abc} \leq \min\{k_{10,ab}, k_{10,bc}\} \leq k_{10}$, similarly $\nu_{abc} \leq k_{10}$, and $p^5 \leq p^4 \ldots \leq 1$, we have $\mathbb{E}_{\geq 1}[Y_1] \leq \max\{1, p^3 k_{10}\}$. $\mathbb{E}_{\geq 1}[Y_{10}] \leq \mathbb{E}[Y_{10}] = p^6 N_{10}$ implies

$$p \geq \max\{\sqrt[6]{1/N_{10}}, \sqrt[3]{k_{10}/N_{10}}\}. \tag{9}$$

Choose $\epsilon_{KV} \geq 0$ and let $\epsilon_{KV}\mathbb{E}[Y_{10}] = a_6\sqrt{\mathbb{E}[Y_{10}]\mathbb{E}_{\geq 1}[Y_{10}]}\lambda^6$. Applying Theorem 2 to Y_{10} given (6) and (9), the right hand side of (8) is $\mathcal{O}(\exp(-\gamma\log m)) = \mathcal{O}(1/m^\gamma)$. Therefore, the error of the 4-clique estimator X_{10} is

$$\delta X_{10} = \frac{1}{p^6}\delta Y_{10} = \frac{1}{p^6}(\epsilon_{KV}p^6 N_{10}) = \epsilon N_{10}$$

with probability greater than $1 - \frac{1}{m^\gamma}$. □

Now we are ready to prove the corollary by comparing Theorem 1 and Lemma 1. Fix $p, \delta, > 0$ and $\gamma > 1$ such that $p = \Omega(1/\log m)$ and $\delta = m^{-\gamma} = \Omega(1/m)$. Now we analyze the bounds ϵ_{KV} and ϵ_{RK}. For any graph and a_6 defined in Theorem 2,

$$\frac{1}{a_6^2} \leq 1, \quad \frac{\gamma}{(5+\gamma)^{12}} \leq 1, \quad \log(2^{1/\gamma}m) \leq 2^{1/\gamma}\log m \leq 2\log m, \quad \left(\frac{k_{10}}{N_{10}}\right)^{2/3} \leq 1. \tag{10}$$

We further require $k_{10} \leq N_{10}^{5/6}$. Then condition on p with 10 implies

$$p^{11} \geq 1/\log^{11}(m) \geq \frac{\gamma\log(2^{1/\gamma}m)}{2a_6^2(5+\gamma)^{12}\log^{12}(m)}\min\left\{k_{10}/N_{10}^{5/6}, (k_{10}/N_{10})^{2/3}\right\}.$$

Rearranging terms,

$$\frac{a_6^2\log^{12}(m^{5+\gamma})\max\left\{\sqrt[6]{1/N_{10}}, \sqrt[3]{k_{10}/N_{10}}\right\}}{p} \geq \frac{\log(2m^\gamma)k_{10}/N_{10}}{2p^{12}} \Rightarrow \epsilon_{KV}^2 \geq \epsilon_{RK}^2.$$

We note that in addition to the favorable scaling, the looseness of inequalities in (10) also favors the use of Theorem 1 over Lemma 1.

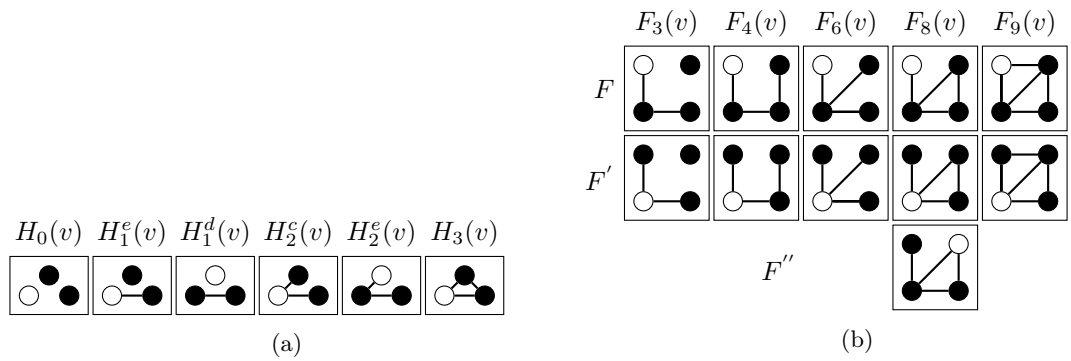


Figure 6: Unique (a) 3-subgraphs and (b) 4-subgraphs from the perspective of the white vertex is v. F_8 is the only subgraph with a third vertex automorphism $F_8^{''}$ because no other subgraph contains vertices with 3 different degrees.

A.3 Additional equations

Preliminary scatter equations:

$$\begin{aligned} n_{3,va} &= |\Gamma(v) \cap \Gamma(a)|, \qquad n^e_{1,va} = |V| - (|\Gamma(v)| + |\Gamma(a)| - |\Gamma(v) \cap \Gamma(a)|), \\ n^c_{2,va} &= |\Gamma(a)| - |\Gamma(v) \cap \Gamma(a)| - 1, \qquad n^e_{2,va} = |\Gamma(v)| - |\Gamma(v) \cap \Gamma(a)| - 1. \end{aligned} \tag{11}$$

Edge pivot equations for both local and global 4-profile:

$$\begin{aligned} \sum_{a\in\Gamma(v)} \binom{n^e_{1,va}}{2} &= F_1(v) + F_2(v), \qquad \sum_{a\in\Gamma(v)} \binom{n^c_{2,va}}{2} = 3F_6^{'}(v) + F_8^{'}(v) \\ \sum_{a\in\Gamma(v)} \binom{n_{3,va}}{2} &= F_9^{'}(v) + 3F_{10}(v), \qquad \sum_{a\in\Gamma(v)} n^e_{1,va} n^c_{2,va} = 2F_3^{'}(v) + F_4^{'}(v) \\ \sum_{a\in\Gamma(v)} n^e_{1,va} n_{3,va} &= 2F_5(v) + F_8^{''}(v), \qquad \sum_{a\in\Gamma(v)} n^c_{2,va} n^e_{2,va} = F_4^{'}(v) + 2F_7(v) \\ \sum_{a\in\Gamma(v)} n^c_{2,va} n_{3,va} &= 2F_8^{'}(v) + 2F_9^{'}(v), \quad n^d_{1,v}|\Gamma(v)| = F_2(v) + F_4(v) + F_8(v) \end{aligned} \tag{12}$$

Edge pivot equations for local 4-profile only (note the last 2 equations require calculating the local 3-profile):

$$\begin{aligned} \sum_{a\in\Gamma(v)} \binom{n^e_{2,va}}{2} &= F_6(v) + F_8(v), \qquad \sum_{a\in\Gamma(v)} n^e_{1,va} n^e_{2,va} = F_3(v) + F_4(v) \\ \sum_{a\in\Gamma(v)} n^e_{2,va} n_{3,va} &= F_8^{''}(v) + 2F_9(v), \qquad \sum_{a\in\Gamma(v)} n_{3,a} - n_{3,va} = F_8(v) + 2F_9(v) + 3F_{10}(v) \\ \sum_{a\in\Gamma(v)} n^e_{2,a} - n^c_{2,va} &= F_4(v) + 2F_7(v) + F_8^{''}(v) + 2F_9^{'}(v) \end{aligned} \tag{13}$$

Equations for directly counting $F_{10}(v)$ and $F_8(v)$:

$$\begin{aligned} \sum_{a\in\Gamma(v)} |(b,c) \in \Delta(a) : b \in \Gamma(v), c \in \Gamma(v)| &= 3F_{10}(v) \\ \sum_{a\in\Gamma(v)} |(b,c) \in \Delta(a) : b \notin \Gamma(v), c \notin \Gamma(v)| &= F_8(v) \end{aligned} \tag{14}$$

Normalization equation (across all automorphisms): $\sum_i^{17} F_i(v) = \binom{|V|-1}{3}$.

To calculate the global 4-profile, we utilize global symmetry equations and scaling. Let $F_i = \sum_{v \in V} F_i(v)$. Globally, each subgraph count is in exact proportion with the same subgraph counted from a different vertex automorphism. The ratio depends on the subgraph's degree distribution:

$$F_3 = 2F_3^{'}, \quad F_4 = F_4^{'}, \quad F_6 = 3F_6^{'}, \quad F_8 = F_8^{'}, \quad F_8^{''} = 2F_8, \quad F_9 = F_9^{'} \tag{15}$$

Global symmetry makes the equation for F_8 and the system (13) linearly dependent. We sum across vertices, invert a single 11×11 system to yield the final global 4-profile $[N_0, \ldots N_{10}^T]$ by scaling appropriately:

$$\begin{aligned} &N_0 = \frac{F_0}{4}, \quad N_1 = \frac{F_1}{2}, \quad N_2 = \frac{F_2}{4}, \quad N_3 = F_3^{'}, \quad N_4 = \frac{F_4^{'}}{2}, \quad N_5 = \frac{F_5}{3}, \\ &N_6 = F_6^{'}, \quad N_7 = \frac{F_7}{4}, \quad N_8 = F_8, \quad N_9 = \frac{F_9}{2}, \quad N_{10} = \frac{F_{10}}{4} \end{aligned} \tag{16}$$

A.4 Analysis of 2-hop histogram

Next, we upper bound savings from our 2-hop histogram by analyzing the improvement when only the information transmitted across the network to a vertex v is each non-neighboring vertex and its final count $\oplus_{a \in \Gamma(v)} c[p]$. Let $h_v = |\Gamma(\Gamma(v)) \setminus \{\Gamma(v) \cup v\}|$. For each v, the difference between full and histogram information is at most $\sum_{a \in \Gamma(v)} (|\Gamma(a)| - 1) - 2h_v$. The exact benefit of (5) depends on the internal implementation of the reduce operation $\oplus$ as pairs of neighbors are gathered.

Counting the number of distinct pairs of 2-paths to each 2-hop neighbor, *i.e.* $\frac{1}{2}(c[p]^2 - c[p])$, requires counting the second moment of c taken over h_v terms. Due to a result by Alon ([27], Proposition 3.7), the memory required to count this value exactly (moreover, to approximate it deterministically) is $\Omega(h_v)$. Thus, up to implementation details, our memory use is optimal.

A.5 Implementation details

To improve the practical performance of 4-Prof-Dist (see Algorithm 1 for pseudocode), we handle low and high degree vertices differently. As in GraphLab PowerGraph's standard triangle counting, cuckoo hash tables are used if the vertex degree is above a threshold. Now, we also threshold vertices to determine whether the 2-hop histogram in Section 3.3 will be either a vector or an unordered map. This is because sorting and merging operations on a vector scale poorly with increasing degree size, while an unordered map has constant lookup time. We found that this approach successfully trades off processing time and memory consumption.

A.6 Extension to 4-profile sparsifier

Another advantage to read-k function families is that they are simpler to extend to more complex subgraphs. We now state concentration results for the full 4-profile sparsifier evaluated experimentally in Section 4

The edge sampling matrix $\mathbf{H}$ is defined by the relations

$$\begin{bmatrix} \mathbb{E}[Y_0] \\ \vdots \\ \mathbb{E}[Y_{10}] \end{bmatrix} = \mathbf{H} \begin{bmatrix} N_0 \\ \vdots \\ N_{10} \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} X_0 \\ \vdots \\ X_{10} \end{bmatrix} = \mathbf{H}^{-1} \begin{bmatrix} Y_0 \\ \vdots \\ Y_{10} \end{bmatrix}, \qquad \text{where}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 1-p & (1-p)^2 & (1-p)^2 & (1-p)^3 & (1-p)^3 & (1-p)^3 & (1-p)^4 & (1-p)^4 & (1-p)^5 & (1-p)^6 \\ 0 & p & 2p(1-p) & 2p(1-p) & 3p(1-p)^2 & 3p(1-p)^2 & 3p(1-p)^2 & 4p(1-p)^3 & 4p(1-p)^3 & 5p(1-p)^4 & 6p(1-p)^5 \\ 0 & 0 & p^2 & 0 & p^2(1-p) & 0 & 0 & 2p^2(1-p)^2 & p^2(1-p)^2 & 2p^2(1-p)^3 & 3p^2(1-p)^4 \\ 0 & 0 & 0 & p^2 & 2p^2(1-p) & 3p^2(1-p) & 3p^2(1-p) & 4p^2(1-p)^2 & 5p^2(1-p)^2 & 8p^2(1-p)^3 & 12p^2(1-p)^4 \\ 0 & 0 & 0 & 0 & p^3 & 0 & 0 & 4p^3(1-p) & 2p^3(1-p) & 6p^3(1-p)^2 & 12p^3(1-p)^3 \\ 0 & 0 & 0 & 0 & 0 & p^3 & 0 & 0 & p^3(1-p) & 2p^3(1-p)^2 & 4p^3(1-p)^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & p^3 & 0 & p^3(1-p) & 2p^3(1-p)^2 & 4p^3(1-p)^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & p^4 & 0 & p^4(1-p) & 3p^4(1-p)^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p^4 & 4p^4(1-p) & 12p^4(1-p)^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p^5 & 6p^5(1-p) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & p^6 \end{bmatrix}$$

Let $t = \frac{p-1}{p}$. Then the inverse sampling matrix is given by

$$\mathbf{H}^{-1} = \begin{bmatrix} 1 & t & t^2 & t^2 & t^3 & t^3 & t^3 & t^4 & t^4 & t^5 & t^6 \\ 0 & \frac{1}{p} & \frac{2t}{p} & \frac{2t}{p} & \frac{3t^2}{p} & \frac{3t^2}{p} & \frac{3t^2}{p} & \frac{4t^3}{p} & \frac{4t^3}{p} & \frac{5t^4}{p} & \frac{6t^5}{p} \\ 0 & 0 & \frac{1}{p^2} & 0 & \frac{t}{p^2} & 0 & 0 & \frac{2t^2}{p^2} & \frac{t^2}{p^2} & \frac{2t^3}{p^2} & \frac{3t^4}{p^2} \\ 0 & 0 & 0 & \frac{1}{p^2} & \frac{2t}{p^2} & \frac{3t}{p^2} & \frac{3t}{p^2} & \frac{4t^2}{p^2} & \frac{5t^2}{p^2} & \frac{8t^3}{p^2} & \frac{12t^4}{p^2} \\ 0 & 0 & 0 & 0 & \frac{1}{p^3} & 0 & 0 & \frac{4t}{p^3} & \frac{2t}{p^3} & \frac{6t^2}{p^3} & \frac{12t^3}{p^3} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{p^3} & 0 & 0 & \frac{t}{p^3} & \frac{2t^2}{p^3} & \frac{4t^3}{p^3} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{p^3} & 0 & \frac{t}{p^3} & \frac{2t^2}{p^3} & \frac{4t^3}{p^3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{p^4} & 0 & \frac{t}{p^4} & \frac{3t^2}{p^4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{p^4} & \frac{4t}{p^4} & \frac{12t^2}{p^4} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{p^5} & \frac{6t}{p^5} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{p^6} \end{bmatrix} \tag{17}$$

The binomial coefficients in these matrices influence our concentration bounds, which we now state:

Theorem 3 (4-profile sparsifier estimators)**.** *Consider the sampling process described above and in Section 2. Let $X_i,\ 0 \le i \le 10$ (and $\mathbf{X}$ be a vector of these estimates) be the actual estimates of 4-profiles. Let k_i be the maximum number of subgraphs F_i sharing a common edge. Let $Y_i,\ 0 \le i \le 10$ be the 4 profile counts of the sparsified graph. Then, $N_i,\ 0 \le i \le 10$ be the actual counts. Let $C = 36864/2$. If*

$$p \ge \left(\frac{C\log(2/\delta)k_{10}}{\epsilon^2 N_{10}}\right)^{1/12}, \quad p \ge \left(\frac{C\log(2/\delta)(k_9+6k_{10})}{\epsilon^2(N_9+6N_{10})}\right)^{1/10}, \quad p \ge \left(\frac{C\log(2/\delta)(k_8+4k_9+12k_{10})}{\epsilon^2(N_8+4N_9+12N_{10})}\right)^{1/8}$$

$$p \ge \left(\frac{C\log(2/\delta)(k_7+k_9+3k_{10})}{\epsilon^2(N_7+N_9+3N_{10})}\right)^{1/8}, \quad p \ge \left(\frac{C\log(2/\delta)(k_6+k_7+2k_9+4k_{10})}{\epsilon^2(N_6+N_8+2N_9+4N_{10})}\right)^{1/6}$$

$$p \ge \left(\frac{C\log(2/\delta)(k_5+k_7+2k_9+4k_{10})}{\epsilon^2(N_5+N_8+2N_9+4N_{10})}\right)^{1/6}, \quad p \ge \left(\frac{C\log(2/\delta)(k_4+4k_7+2k_8+6k_9+12k_{10})}{\epsilon^2(N_4+4N_7+2N_8+6N_9+12N_{10})}\right)^{1/6}$$

$$p \ge \left(\frac{C\log(2/\delta)(k_3+2k_4+3k_5+3k_6+4k_7+5k_8+8k_9+12k_{10})}{\epsilon^2(N_3+2N_4+3N_5+3N_6+4N_7+5N_8+8N_9+12N_{10})}\right)^{1/4}$$

$$p \ge \left(\frac{C\log(2/\delta)(k_2+k_4+2k_7+k_8+2k_9+3k_{10})}{\epsilon^2(N_2+N_4+2N_7+N_8+2N_9+3N_{10})}\right)^{1/4}$$

$$p \ge \left(\frac{C\log(2/\delta)(k_1+2k_2+2k_3+3k_4+3k_5+3k_6+4k_7+4k_8+5k_9+6k_{10})}{\epsilon^2(N_1+2N_2+2N_3+3N_4+3N_5+3N_6+4N_7+4N_8+5N_9+6N_{10})}\right)^{1/2}$$

$$n_0 \le |V|^2\left(|V|^2 - \frac{C\log(2/\delta)}{\epsilon^2}\right)$$

then $\|\delta\mathbf{X}\|_\infty \le \epsilon\binom{|V|}{4}$.

Proof. We apply Proposition 1 a total of 11 times to the sampling-estimator system defined above by $\mathbf{H}$ and $\mathbf{H}^{-1}$. In our context, each sampled subgraph count Y_i is a sum of functions in a read-k_i family, where $k_i \le \min\{|V|-2, N_i\}$. Let $s_{i,e}$ be the maximum number of subgraphs F_i sharing a common edge e, and let

$s_i = \max_e s_{i,e}$, for $i = 0, \ldots, 10$. The Y_i's have the following parameters:

$$
\begin{aligned}
r_{Y_0} &= \binom{|V|}{4}, \quad k_{Y_0} = |V| \\
r_{Y_1} &= N_1 + 2N_2 + 2N_3 + 3N_4 + 3N_5 + 3N_6 + 4N_7 + 4N_8 + 5N_9 + 6N_{10} \\
k_{Y_1} &= s_1 + 2s_2 + 2s_3 + 3s_4 + 3s_5 + 3s_6 + 4s_7 + 4s_8 + 5s_9 + 6s_{10} \\
r_{Y_2} &= N_2 + N_4 + 2N_7 + N_8 + 2N_9 + 3N_{10} \\
k_{Y_2} &= s_2 + s_4 + 2s_7 + s_8 + 2s_9 + 3s_{10} \\
r_{Y_3} &= N_3 + 2N_4 + 3N_5 + 3N_6 + 4N_7 + 5N_8 + 8N_9 + 12N_{10} \\
k_{Y_3} &= s_3 + 2s_4 + 3s_5 + 3s_6 + 4s_7 + 5s_8 + 8s_9 + 12s_{10} \\
r_{Y_4} &= N_4 + 4N_7 + 2N_8 + 6N_9 + 12N_{10}, \quad k_{Y_4} = s_4 + 4s_7 + 2s_8 + 6s_9 + 12s_{10} \\
r_{Y_5} &= N_5 + N_8 + 2N_9 + 4N_{10}, \quad k_{Y_5} = s_5 + s_8 + 2s_9 + 4s_{10} \\
r_{Y_6} &= N_6 + N_8 + 2N_9 + 4N_{10}, \quad k_{Y_6} = s_6 + s_8 + 2s_9 + 4s_{10} \\
r_{Y_7} &= N_7 + N_9 + 3N_{10}, \quad k_{Y_7} = s_7 + s_9 + 3s_{10} \\
r_{Y_8} &= N_8 + 4N_9 + 12N_{10}, \quad k_{Y_8} = s_8 + 4s_9 + 12s_{10} \\
r_{Y_9} &= N_9 + 6N_{10}, \quad k_{Y_9} = s_9 + 6s_{10} \\
r_{Y_{10}} &= N_{10}, \quad k_{Y_{10}} = s_{10}
\end{aligned}
\tag{18}
$$

We show the application of Proposition 1 for Y_7 through Y_9 because Y_{10} was shown in the proof of Theorem 1 and the other cases are similar:

$$
\begin{aligned}
\mathbb{P}\left(|Y_7 - (p^4 N_7 + p^4(1-p)N_9 + 3p^4(1-p)^2 N_{10})| \geq p^4\epsilon(N_7 + N_9 + 3N_{10})\right) &\leq 2\exp\left(-\frac{2p^8\epsilon^2(N_7 + N_9 + 3N_{10})}{s_7 + s_9 + 3s_{10}}\right) \\
\mathbb{P}\left(|Y_8 - (p^4 N_8 + 4p^4(1-p)N_9 + 12p^4(1-p)^2 N_{10})| \geq p^4\epsilon(N_8 + 4N_9 + 12N_{10})\right) &\leq 2\exp\left(-\frac{2p^8\epsilon^2(N_8 + 4N_9 + 12N_{10})}{s_8 + 4s_9 + 12s_{10}}\right) \\
\mathbb{P}\left(|Y_9 - (p^5 N_9 + 6p^5(1-p)N_{10})| \geq \epsilon(N_9 + 6N_{10})\right) &\leq 2\exp\left(-\frac{2\epsilon^2(N_9 + 6N_{10})}{s_9 + 6s_{10}}\right) \\
\Rightarrow \mathbb{P}\left(|\frac{1}{p^5}Y_9 - (N_9 + 6(1-p)N_{10})| \geq \epsilon(N_9 + 6N_{10})\right) &\leq 2\exp\left(-\frac{2p^{10}\epsilon^2(N_9 + 6N_{10})}{s_9 + 6s_{10}}\right) \\
\mathbb{P}\left(|Y_{10} - p^6 N_{10}| \geq \epsilon N_{10}\right) &\leq 2\exp\left(-\frac{2\epsilon^2 N_{10}}{s_{10}}\right) \\
\Rightarrow \mathbb{P}\left(|X_{10} - N_{10}| \geq \epsilon N_{10}\right) = \mathbb{P}\left(|Y_3 - p^6 N_{10}| \geq p^6\epsilon N_{10}\right) &\leq 2\exp\left(-\frac{2p^{12}\epsilon^2 N_{10}}{s_{10}}\right)
\end{aligned}
$$

Rearranging to solve for p, we have

$$
\begin{aligned}
p \geq \left(\frac{\log(2/\delta)s_{10}}{2\epsilon^2 N_{10}}\right)^{1/12}, \quad p \geq \left(\frac{\log(2/\delta)(s_9 + 6s_{10})}{2\epsilon^2(N_9 + 6N_{10})}\right)^{1/10}, \quad p \geq \left(\frac{\log(2/\delta)(s_8 + 4s_9 + 12s_{10})}{2\epsilon^2(N_8 + 4N_9 + 12N_{10})}\right)^{1/8} \\
p \geq \left(\frac{\log(2/\delta)(s_7 + s_9 + 3s_{10})}{2\epsilon^2(N_7 + N_9 + 3N_{10})}\right)^{1/8}, \quad p \geq \left(\frac{\log(2/\delta)(s_6 + s_7 + 2s_9 + 4s_{10})}{2\epsilon^2(N_6 + N_8 + 2N_9 + 4N_{10})}\right)^{1/6} \\
p \geq \left(\frac{\log(2/\delta)(s_5 + s_7 + 2s_9 + 4s_{10})}{2\epsilon^2(N_5 + N_8 + 2N_9 + 4N_{10})}\right)^{1/6}, \quad p \geq \left(\frac{\log(2/\delta)(s_4 + 4s_7 + 2s_8 + 6s_9 + 12s_{10})}{2\epsilon^2(N_4 + 4N_7 + 2N_8 + 6N_9 + 12N_{10})}\right)^{1/6} \\
p \geq \left(\frac{\log(2/\delta)(s_3 + 2s_4 + 3s_5 + 3s_6 + 4s_7 + 5s_8 + 8s_9 + 12s_{10})}{2\epsilon^2(N_3 + 2N_4 + 3N_5 + 3N_6 + 4N_7 + 5N_8 + 8N_9 + 12N_{10})}\right)^{1/4} \\
p \geq \left(\frac{\log(2/\delta)(s_2 + s_4 + 2s_7 + s_8 + 2s_9 + 3s_{10})}{2\epsilon^2(N_2 + N_4 + 2N_7 + N_8 + 2N_9 + 3N_{10})}\right)^{1/4} \\
p \geq \left(\frac{\log(2/\delta)(s_1 + 2s_2 + 2s_3 + 3s_4 + 3s_5 + 3s_6 + 4s_7 + 4s_8 + 5s_9 + 6s_{10})}{2\epsilon^2(N_1 + 2N_2 + 2N_3 + 3N_4 + 3N_5 + 3N_6 + 4N_7 + 4N_8 + 5N_9 + 6N_{10})}\right)^{1/2}
\end{aligned}
\tag{19}
$$

The final condition comes from the result for Y_0:

$$n_0 \leq \binom{|V|}{4} - \frac{\log(2/\delta)|V|^2}{2\epsilon^2} \leq |V|^2 \left(|V|^2 - \frac{\log(2/\delta)}{2\epsilon^2}\right) \tag{20}$$

Plugging into our estimators (given by $\mathbf{H}^{-1}$):

$$\begin{aligned}
\delta X_0 &\leq \epsilon(n_1 + n_2 + n_3) + \epsilon(n_1 + 2n_2 + 3n_3 + n_2 + 3n_3 + n_3) \\
&\leq \epsilon(2n_1 + 4n_2 + 8n_3) \leq 8\epsilon\binom{|V|}{3} \\
\delta X_1 &\leq \epsilon(N_1 + 2N_2 + 2N_3 + 3N_4 + 3N_5 + 3N_6 + 4N_7 + 4N_8 + 5N_9 + 6N_{10}) + 2\epsilon(N_2 + N_4 + 2N_7 + N_8 + 2N_9 + 3N_{10}) \\
&\quad + 2\epsilon(N_3 + 2N_4 + 3N_5 + 3N_6 + 4N_7 + 5N_8 + 8N_9 + 12N_{10}) + 3\epsilon(N_4 + 4N_7 + 2N_8 + 6N_9 + 12N_{10}) \\
&\quad + 3\epsilon(N_5 + N_8 + 2N_9 + 4N_{10}) + 3\epsilon(N_6 + N_8 + 2N_9 + 4N_{10}) + 4\epsilon(N_7 + N_9 + 3N_{10}) \\
&\quad + 4\epsilon(N_8 + 4N_9 + 12N_{10}) + 5\epsilon(N_9 + 6N_{10}) + 6\epsilon(N_{10}) \\
&\leq \epsilon(N_1 + \ldots + 192N_{10}) \leq 192\epsilon\binom{|V|}{4} \\
\delta X_2 &\leq \epsilon(N_2 + N_4 + 2N_7 + N_8 + 2N_9 + 3N_{10}) + \epsilon(N_4 + 4N_7 + 2N_8 + 6N_9 + 12N_{10}) + 2\epsilon(N_7 + N_9 + 3N_{10}) \\
&\quad + \epsilon(N_8 + 4N_9 + 12N_{10}) + 2\epsilon(N_9 + 6N_{10}) + 3\epsilon(N_{10}) \\
&\leq \epsilon(N_2 + \ldots + 48N_{10}) \leq 48\epsilon\binom{|V|}{4} \\
\delta X_3 &\leq \epsilon(N_3 + 2N_4 + 3N_5 + 3N_6 + 4N_7 + 5N_8 + 8N_9 + 12N_{10}) + 2\epsilon(N_4 + 4N_7 + 2N_8 + 6N_9 + 12N_{10}) \\
&\quad + 3\epsilon(N_5 + N_8 + 2N_9 + 4N_{10}) + 3\epsilon(N_6 + N_8 + 2N_9 + 4N_{10}) + 4\epsilon(N_7 + N_9 + 3N_{10}) + 5\epsilon(N_8 + 4N_9 + 12N_{10}) \\
&\quad + 8\epsilon(N_9 + 6N_{10}) + 12\epsilon(N_{10}) \\
&\leq \epsilon(N_3 + 4N_4 + 6N_5 + \ldots + 192N_{10}) \leq 192\epsilon\binom{|V|}{4} \\
\delta X_4 &\leq \epsilon(N_4 + 4N_7 + 2N_8 + 6N_9 + 12N_{10}) + 4\epsilon(N_7 + N_9 + 3N_{10}) + 2\epsilon(N_8 + 4N_9 + 12N_{10}) + 6\epsilon(N_9 + 6N_{10}) + 12\epsilon(N_{10}) \\
&\leq \epsilon(N_4 + \ldots + 96N_{10}) \leq 96\epsilon\binom{|V|}{4} \\
\delta X_5 &\leq \epsilon(N_5 + N_8 + 2N_9 + 4N_{10}) + \epsilon(N_8 + 4N_9 + 12N_{10}) + 2\epsilon(N_9 + 6N_{10}) + 4\epsilon(N_{10}) \\
&\leq \epsilon(N_5 + \ldots + 32N_{10}) \leq 32\epsilon\binom{|V|}{4} \\
\delta X_6 &\leq \epsilon(N_6 + N_8 + 2N_9 + 4N_{10}) + \epsilon(N_8 + 4N_9 + 12N_{10}) + 2\epsilon(N_9 + 6N_{10}) + 4\epsilon(N_{10}) \\
&\leq \epsilon(N_7 + 2N_9 + 32N_{10}) \leq 32\epsilon\binom{|V|}{4} \\
\delta X_7 &\leq \epsilon(N_7 + N_9 + 3N_{10}) + \epsilon(N_9 + 6N_{10}) + 3\epsilon(N_{10}) \\
&\leq \epsilon(N_7 + 2N_9 + 12N_{10}) \leq 12\epsilon\binom{|V|}{4} \\
\delta X_8 &\leq \epsilon(N_8 + 4N_9 + 12N_{10}) + 4\epsilon(N_9 + 6N_{10}) + 12\epsilon(N_{10}) \\
&\leq \epsilon(N_8 + 8N_9 + 48N_{10}) \leq 48\epsilon\binom{|V|}{4} \\
\delta X_9 &\leq \epsilon(N_9 + 6N_{10}) + 6\epsilon(N_{10}) \\
&\leq \epsilon(N_9 + 12N_{10}) \leq 12\epsilon\binom{|V|}{4} \\
\delta X_{10} &\leq \epsilon N_{10}.
\end{aligned}$$

Thus the maximum deviation in any estimator is less than $192\epsilon\binom{|V|}{4}$. Substituting $\tilde{\epsilon} = \epsilon/192$ completes the proof.

□