4. EXPERIMENTS

Let us now describe the implementation and experimental results of our algorithm. We implement 4-Prof-Dist on GraphLab v2.2 (PowerGraph) [12] and measure its running time and accuracy on large input graphs.[3] First, we show that edge sampling yields very good approximation results for global 4-profile counts and achieves substantial execution speedups and network traffic savings 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 [13], which we use as a baseline for our results. Note also that Orca produces only a partial 4-subgraph count, *i.e.* it calculates only connected 4-subgraphs, while 4-Prof-Dist calculates all 17 per vertex.

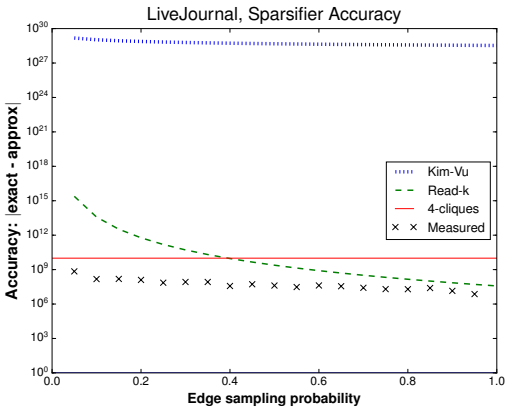


Figure 5: Comparison of 4-clique sparsifier concentration bounds with accuracy measured in edge sampling experiments on the LiveJournal graph.

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.[4] 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) [18]. 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.

4.1 Results

Accuracy: The first result is that our edge sampling approach greatly improves running time while maintaining a very good approximation of the *global* 4-profile. In Figure 6a we can see that the running time decreases drastically when the sampling probability decreases. At the same time, Figure 6b shows that the mean ratio of true to estimated global 4-profiles is within $\pm 2.5\%$. Similar to [15], which uses a more complex sampling scheme to count connected 4-subgraphs, this ratio is usually much less than 1%. 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 5 we compare theoretical concentration bounds on a logarithmic scale and show the benefit of Theorem 2. While the guarantees provided by Kim-Vu [16] 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 for large sampling probabilities ($p \geq 0.5$), the measured error is at most 2 orders of magnitude smaller than the value predicted by Theorem 2.

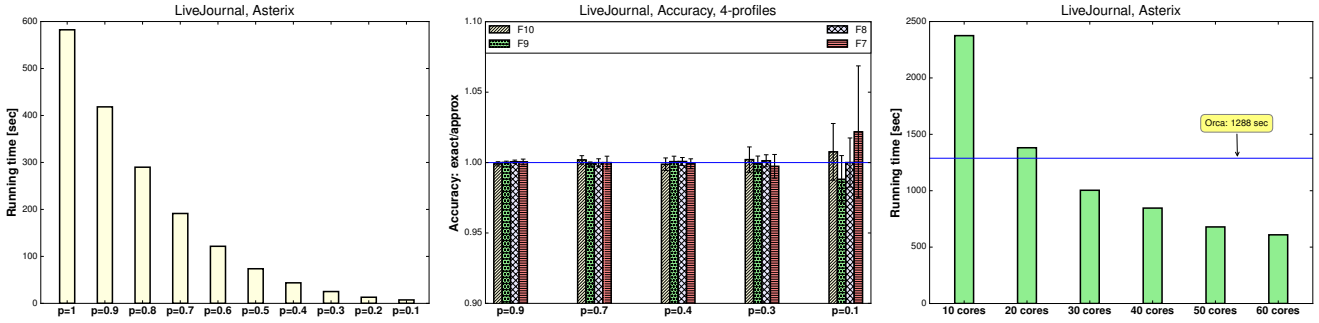
2-hop histogram: Now we compare two methods of calculating the left hand side of (5) from Section 2.3. We show that a simple implementation in which a vertex gathers its full 2-hop neighborhood (*i.e.*, IDs of its neighbors' neighbors) is much less efficient than the *two-hop histogram* approach used in 4-Prof-Dist (see Section 2.3). In Figures 7 and 9 we can see that the histogram approach is an order of magnitude faster for various numbers of machines, and that its network requirements are up to 5x less than that of the simple implementation. Moreover, our algorithm could handle much larger graphs while the simple implementation ran out of memory.

Running time: Finally, we show that 4-Prof-Dist can run much faster than the 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 6c 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 [13]. Our 4-Prof-Dist algorithm 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 8 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 [20], 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 8 for $p = 1$). We also note a substantial running time benefit of the sampling approach for *global* 4-profiles. In Figures 8 and 10, we see that with $p = 0.1$ we can achieve order of magnitude improvements in both speed and network traffic. This sampling probability maintains very good accuracy, as shown in Figure 6b.

5. CONCLUSIONS

We introduced a novel distributed algorithm for estimating 4-profiles of large graphs. We relied on two theoretical results that can be of independent interest: that 4-profiles can be estimated with limited 2-hop information and that randomly erasing edges gives sharper approximation compared to previous analysis. We showed that our scheme outperforms the previous state of the art and can exploit cloud infrastructure to scale.

[3]Available at `http://github.com/eelenberg/4-profiles`
[4]Amazon Web Services - `http://aws.amazon.com`



(a) (b) (c)

Figure 6: 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 to the estimated count. (c) – Comparison of running times of Orca and our exact 4-Prof-Dist algorithm. Clearly, 4-Prof-Dist benefits from the use of multiple cores.

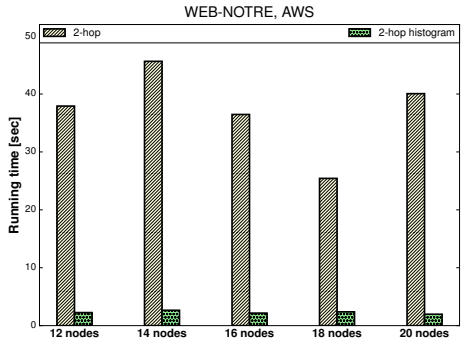


Figure 7: AWS cluster of up to 20 machines (nodes), results averaged over 10 iterations. Running time comparing naive 2-hop implementation and 2-hop histogram approach on the Notre Dame web graph.

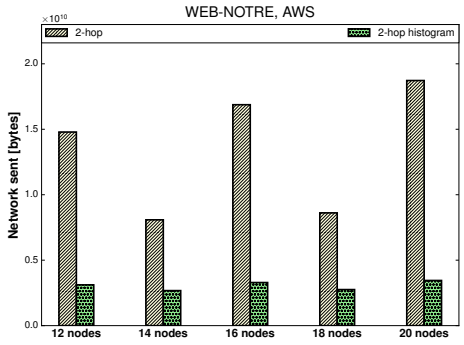


Figure 9: Network usage comparing naive 2-hop implementation and 2-hop histogram approach on the Notre Dame web graph.

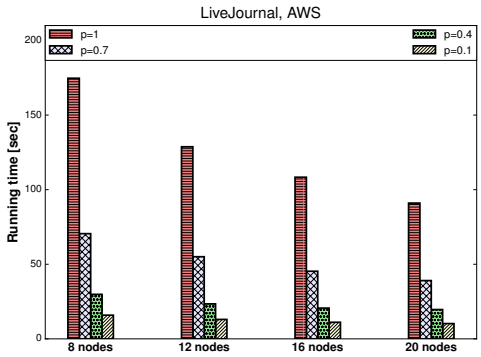


Figure 8: Running time of 4-Prof-Dist for various number of compute nodes and sampling probability p, on the LiveJournal graph.

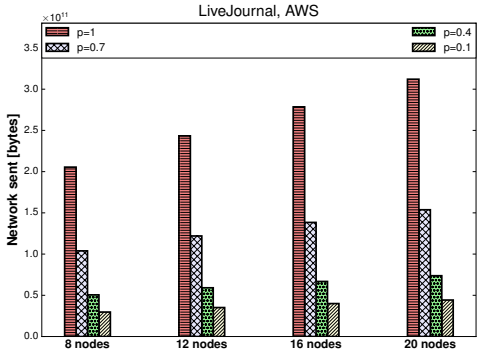


Figure 10: Network usage of 4-Prof-Dist for various number of compute nodes and sampling probability p, on the LiveJournal graph.

Acknowledgements: We would like to thank the anonymous reviewers for their useful comments. This research has been supported by NSF Grants CCF 1344179, 1344364, 1407278, 1422549 and ARO YIP W911NF-14-1-0258.

6. REFERENCES

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

[2] N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield. Efficient Graphlet Counting for Large Networks. In *IEEE International Conference on Data Mining*, 2015.

[3] N. Alon, Y. Matias, and M. Szegedy. The Space Complexity of Approximating the Frequency Moments. In *STOC*, pages 20–29, 1996.

[4] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient Semi-Streaming Algorithms for Local Triangle Counting in Massive Graphs. In *KDD*, 2008.

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

[6] T. Eden, A. Levi, D. Ron, and C. Seshadhri. Approximately Counting Triangles in Sublinear Time. In *FOCS*, pages 614–633, 2015.

[7] 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*, pages 229–238, 2015.

[8] E. R. Elenberg, K. Shanmugam, M. Borokhovich, and A. G. Dimakis. Distributed Estimation of Graph 4-profiles. `http://arxiv.org/abs/1510.02215`, 2015.

[9] F. Fei, B. Jie, and D. Zhang. Frequent and Discriminative Subnetwork Mining for Mild Cognitive Impairment Classification. *Brain Connectivity*, 4(5):347–360, June 2014.

[10] I. Finocchi, M. Finocchi, and E. G. Fusco. Clique Counting in MapReduce: Algorithms and Experiments. *ACM Journal of Experimental Algorithmics*, 20(1), 2015.

[11] D. Gavinsky, S. Lovett, M. Saks, and S. Srinivasan. A Tail Bound for Read-k Families of Functions. *Random Structures & Algorithms*, 47(1):99–108, 2015.

[12] 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)*, pages 17–30, 2012.

[13] T. Hočevar and J. Demšar. A Combinatorial Approach to Graphlet Counting. *Bioinformatics*, 30(4):559–65, Feb. 2014.

[14] S. Janson, K. Oleszkiewicz, and A. Ruciński. Upper Tails for Subgraph Counts in Random Graphs. *Israel Journal of Mathematics*, 142(1):61–92, 2004.

[15] M. Jha, C. Seshadhri, and A. Pinar. Path Sampling: A Fast and Provable Method for Estimating 4-Vertex Subgraph Counts. In *WWW*, pages 495–505, 2015.

[16] J. H. Kim and V. H. Vu. Concentration of Multivariate Polynomials and Its Applications. *Combinatorica*, 20(3):417–434, 2000.

[17] M. Kowaluk, A. Lingas, and E.-M. Lundell. Counting and Detecting Small Subgraphs via Equations. *SIAM Journal of Discrete Mathematics*, 27(2):892–909, 2013.

[18] J. Leskovec and A. Krevl. SNAP Datasets: Stanford Large Network Dataset Collection. `http://snap.stanford.edu/data`, June 2014.

[19] L. Lovász. *Large Networks and Graph Limits*, volume 60. American Mathematical Soc., 2012.

[20] A. Milinković, S. Milinković, and L. Lazić. A Contribution to Acceleration of Graphlet Counting. In *Infoteh Jahorina Symposium*, volume 14, pages 741–745, 2015.

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

[22] N. Przulj. Biological Network Comparison Using Graphlet Degree Distribution. *Bioinformatics*, 23(2):177–183, 2007.

[23] 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*, pages 979–990, 2014.

[24] T. Schank. *Algorithmic Aspects of Triangle-Based Network Analysis*. PhD thesis, 2007.

[25] 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*, pages 10–18, 2013.

[26] 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*, pages 488–495, 2009.

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

[28] C. E. Tsourakakis, M. Kolountzakis, and G. L. Miller. Triangle Sparsifiers. *Journal of Graph Theory and Applications*, 15(6):703–726, 2011.

[29] J. Ugander, L. Backstrom, M. Park, and J. Kleinberg. Subgraph Frequencies: Mapping the Empirical and Extremal Geography of Large Graph Collections. In *WWW*, pages 1307–1318, 2013.

[30] L. R. Varshney, B. L. Chen, E. Paniagua, D. H. Hall, and D. B. Chklovskii. Structural Properties of the Caenorhabditis elegans Neuronal Network. 7(2), 2011.

[31] V. V. Williams, J. Wang, R. Williams, and H. Yu. Finding Four-Node Subgraphs in Triangle Time. *SODA*, pages 1671–1680, 2014.

APPENDIX

A. 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 2.3 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.

B. EXTENSION TO GLOBAL 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. Using the notation in Section 3, the edge sampling matrix $\mathbf{H}$ is defined by the relations

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

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

$$\mathbf{H}^{-1} = \begin{bmatrix} S_{11} & S_{12} \\ \mathbf{0}_{4\times 7} & S_{22}\end{bmatrix}, \qquad \text{where}$$

$$S_{11} = \begin{bmatrix} 1 & t & t^2 & t^2 & t^3 & t^3 & t^3 \\ 0 & \frac{1}{p} & \frac{2t}{p} & \frac{2t}{p} & \frac{3t^2}{p} & \frac{3t^2}{p} & \frac{3t^2}{p} \\ 0 & 0 & \frac{1}{p^2} & 0 & \frac{t}{p^2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{p^2} & \frac{2t}{p^2} & \frac{3t}{p^2} & \frac{3t}{p^2} \\ 0 & 0 & 0 & 0 & \frac{1}{p^3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{p^3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{p^3} \end{bmatrix},$$

$$S_{12} = \begin{bmatrix} t^4 & t^4 & t^5 & t^6 \\ \frac{4t^3}{p} & \frac{4t^3}{p} & \frac{5t^4}{p} & \frac{6t^5}{p} \\ \frac{2t^2}{p^2} & \frac{t^2}{p^2} & \frac{2t^3}{p^2} & \frac{3t^4}{p^2} \\ \frac{4t^2}{p^2} & \frac{5t^2}{p^2} & \frac{8t^3}{p^2} & \frac{12t^4}{p^2} \\ \frac{4t}{p^3} & \frac{2t}{p^3} & \frac{6t^2}{p^3} & \frac{12t^3}{p^3} \\ 0 & \frac{t}{p^3} & \frac{2t^2}{p^3} & \frac{4t^3}{p^3} \\ 0 & \frac{t}{p^3} & \frac{2t^2}{p^3} & \frac{4t^3}{p^3} \end{bmatrix},$$

$$S_{22} = \begin{bmatrix} \frac{1}{p^4} & 0 & \frac{t}{p^4} & \frac{3t^2}{p^4} \\ 0 & \frac{1}{p^4} & \frac{4t}{p^4} & \frac{12t^2}{p^4} \\ 0 & 0 & \frac{1}{p^5} & \frac{6t}{p^5} \\ 0 & 0 & 0 & \frac{1}{p^6} \end{bmatrix},$$

and $\mathbf{0}_{4\times 7}$ is a 4×7 matrix of zeros.

The binomial coefficients in these matrices influence our concentration bounds. A more detailed proof of the following result may be found in the extended version of this paper [8].

Theorem 3 (4-profile sparsifier). *Consider the sampling process described above and in Section 3. 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 let N_i, $0 \le i \le 10$, be the actual counts. Choose $0 < \delta < 1$ and $\epsilon > 0$. Let $C = (192)^2/2$ and*

$$k_\alpha = k_2 + k_3, \quad k_\beta = k_4 + k_5 + k_6, \quad k_\gamma = k_7 + k_8,$$
$$N_\alpha = N_2 + N_3, \quad N_\beta = N_4 + N_5 + N_6, \quad N_\gamma = N_7 + N_8.$$

If

$$p \ge \left(\frac{C\log(2/\delta)k_{10}}{\epsilon^2 N_{10}}\right)^{1/12}, \quad p \ge \left(\frac{C\log(2/\delta)(k_9+6k_{10})}{\epsilon^2(N_9+6N_{10})}\right)^{1/10}$$
$$p \ge \left(\frac{C\log(2/\delta)(k_8+4k_9+12k_{10})}{\epsilon^2(N_8+4N_9+12N_{10})}\right)^{1/8}$$
$$p \ge \left(\frac{C\log(2/\delta)(k_7+k_9+3k_{10})}{\epsilon^2(N_7+N_9+3N_{10})}\right)^{1/8}$$
$$p \ge \left(\frac{C\log(2/\delta)(k_6+k_8+2k_9+4k_{10})}{\epsilon^2(N_6+N_8+2N_9+4N_{10})}\right)^{1/6}$$
$$p \ge \left(\frac{C\log(2/\delta)(k_5+k_8+2k_9+4k_{10})}{\epsilon^2(N_5+N_8+2N_9+4N_{10})}\right)^{1/6}$$
$$p \ge \left(\frac{C\log(2/\delta)(k_4+4k_7+2k_8+6k_9+12k_{10})}{\epsilon^2(N_4+4N_7+2N_8+6N_9+12N_{10})}\right)^{1/6}$$
$$p \ge \left(\frac{C\log(2/\delta)}{\epsilon^2}\right)^{1/4} \times \left(\frac{k_3+2k_4+3k_5+3k_6+4k_7+5k_8+8k_9+12k_{10}}{N_3+2N_4+3N_5+3N_6+4N_7+5N_8+8N_9+12N_{10}}\right)^{1/4}$$
$$p \ge \left(\frac{C\log(2/\delta)(k_2+k_4+2k_7+k_8+2k_9+3k_{10})}{\epsilon^2(N_2+N_4+2N_7+N_8+2N_9+3N_{10})}\right)^{1/4}$$
$$p \ge \left(\frac{C\log(2/\delta)(k_1+2k_\alpha+3k_\beta+4k_\gamma+5k_9+6k_{10})}{\epsilon^2(N_1+2N_\alpha+3N_\beta+4N_\gamma+5N_9+6N_{10})}\right)^{1/2}$$
$$n_0 \le |V|^2\left(|V|^2 - \frac{C\log(2/\delta)}{\epsilon^2}\right),$$

then $\|\delta\mathbf{X}\|_\infty \le \epsilon\binom{|V|}{4}$ with probabilty at least $1-\delta$.

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_{Y_i} family, where $k_{Y_i} \le \min\{|V|-2, N_i\}$. Let $k_{i,e}$ be the maximum number of subgraphs F_i sharing a common edge e, and let $k_i = \max_e k_{i,e}$, for $i = 0, \ldots, 10$.

The Y_i's have the following parameters:

$$
\begin{aligned}
r_{Y_0} &= \binom{|V|}{4}, \quad k_{Y_0} = |V| \\
r_{Y_1} &= N_1 + 2N_2 + 2N_3 + 3N_4 + 3N_5 + 3N_6 + 4N_7 + \\
&\quad 4N_8 + 5N_9 + 6N_{10} \\
k_{Y_1} &= k_1 + 2k_2 + 2k_3 + 3k_4 + 3k_5 + 3k_6 + 4k_7 + 4k_8 + \\
&\quad 5k_9 + 6k_{10} \\
r_{Y_2} &= N_2 + N_4 + 2N_7 + N_8 + 2N_9 + 3N_{10} \\
k_{Y_2} &= k_2 + k_4 + 2k_7 + k_8 + 2k_9 + 3k_{10} \\
r_{Y_3} &= N_3 + 2N_4 + 3N_5 + 3N_6 + 4N_7 + 5N_8 + 8N_9 + 12N_{10} \\
k_{Y_3} &= k_3 + 2k_4 + 3k_5 + 3k_6 + 4k_7 + 5k_8 + 8k_9 + 12k_{10} \\
r_{Y_4} &= N_4 + 4N_7 + 2N_8 + 6N_9 + 12N_{10} \\
k_{Y_4} &= k_4 + 4k_7 + 2k_8 + 6k_9 + 12k_{10} \\
r_{Y_5} &= N_5 + N_8 + 2N_9 + 4N_{10}, \quad k_{Y_5} = k_5 + k_8 + 2k_9 + 4k_{10} \\
r_{Y_6} &= N_6 + N_8 + 2N_9 + 4N_{10}, \quad k_{Y_6} = k_6 + k_8 + 2k_9 + 4k_{10} \\
r_{Y_7} &= N_7 + N_9 + 3N_{10}, \quad k_{Y_7} = k_7 + k_9 + 3k_{10} \\
r_{Y_8} &= N_8 + 4N_9 + 12N_{10}, \quad k_{Y_8} = k_8 + 4k_9 + 12k_{10} \\
r_{Y_9} &= N_9 + 6N_{10}, \quad k_{Y_9} = k_9 + 6k_{10} \\
r_{Y_{10}} &= N_{10}, \quad k_{Y_{10}} = k_{10}
\end{aligned}
$$

We apply Proposition 1 to each estimator. This is shown in the proof of Theorem 2 for Y_{10} and in the extended paper [8] for the other estimators. Rearranging to solve for p,

$$
\begin{aligned}
p &\geq \left(\frac{\log(2/\delta)k_{10}}{2\epsilon^2 N_{10}}\right)^{1/12}, \quad p \geq \left(\frac{\log(2/\delta)(k_9 + 6k_{10})}{2\epsilon^2(N_9 + 6N_{10})}\right)^{1/10} \\
p &\geq \left(\frac{\log(2/\delta)(k_8 + 4k_9 + 12k_{10})}{2\epsilon^2(N_8 + 4N_9 + 12N_{10})}\right)^{1/8} \\
p &\geq \left(\frac{\log(2/\delta)(k_7 + k_9 + 3k_{10})}{2\epsilon^2(N_7 + N_9 + 3N_{10})}\right)^{1/8} \\
p &\geq \left(\frac{\log(2/\delta)(k_6 + k_7 + 2k_9 + 4k_{10})}{2\epsilon^2(N_6 + N_8 + 2N_9 + 4N_{10})}\right)^{1/6} \\
p &\geq \left(\frac{\log(2/\delta)(k_5 + k_7 + 2k_9 + 4k_{10})}{2\epsilon^2(N_5 + N_8 + 2N_9 + 4N_{10})}\right)^{1/6} \\
p &\geq \left(\frac{\log(2/\delta)(k_4 + 4k_7 + 2k_8 + 6k_9 + 12k_{10})}{2\epsilon^2(N_4 + 4N_7 + 2N_8 + 6N_9 + 12N_{10})}\right)^{1/6} \\
p &\geq \left(\frac{\log(2/\delta)}{2\epsilon^2}\right)^{1/4} \times \\
&\quad \left(\frac{k_3 + 2k_4 + 3k_5 + 3k_6 + 4k_7 + 5k_8 + 8k_9 + 12k_{10}}{N_3 + 2N_4 + 3N_5 + 3N_6 + 4N_7 + 5N_8 + 8N_9 + 12N_{10}}\right)^{1/4} \\
p &\geq \left(\frac{\log(2/\delta)(k_2 + k_4 + 2k_7 + k_8 + 2k_9 + 3k_{10})}{2\epsilon^2(N_2 + N_4 + 2N_7 + N_8 + 2N_9 + 3N_{10})}\right)^{1/4} \\
p &\geq \left(\frac{\log(2/\delta)(k_1 + 2k_\alpha + 3k_\beta + 4k_\gamma + 5k_9 + 6k_{10})}{2\epsilon^2(N_1 + 2N_\alpha + 3N_\beta + 4N_\gamma + 5N_9 + 6N_{10})}\right)^{1/2},
\end{aligned}
$$

where

$$
\begin{aligned}
&k_\alpha = k_2 + k_3, \quad k_\beta = k_4 + k_5 + k_6, \quad k_\gamma = k_7 + k_8, \\
&N_\alpha = N_2 + N_3, \quad N_\beta = N_4 + N_5 + N_6, \quad N_\gamma = N_7 + N_8.
\end{aligned}
$$

The final condition comes from the result for Y_0:

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

Plugging into our estimators (given by $\mathbf{H}^{-1}$), we get the following error bounds:

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

Thus the maximum deviation in any estimator is less than $192\epsilon\binom{|V|}{4}$. Substituting $\tilde{\epsilon}^2 = \epsilon^2/(192)^2 = \epsilon^2/2C$ completes the proof. □