Beyond Triangles: A Distributed Framework for Estimating 3-profiles of Large Graphs

Ethan R. Elenberg
The University of Texas
Austin, Texas 78712, USA
elenberg@utexas.edu

Michael Borokhovich
The University of Texas
Austin, Texas 78712, USA
michaelbor@utexas.edu

Karthikeyan Shanmugam
The University of Texas
Austin, Texas 78712, USA
karthiksh@utexas.edu

Alexandros G. Dimakis
The University of Texas
Austin, Texas 78712, USA
dimakis@austin.utexas.edu

ABSTRACT
We study the problem of approximating the 3-profile of a large graph. 3-profiles are generalizations of triangle counts that specify the number of times a small graph appears as an induced subgraph of a large graph. Our algorithm uses the novel concept of 3-profile sparsifiers: sparse graphs that can be used to approximate the full 3-profile counts for a given large graph. Further, we study the problem of estimating local and ego 3-profiles, two graph quantities that characterize the local neighborhood of each vertex of a graph.

Our algorithm is distributed and operates as a vertex program over the GraphLab framework. We introduce the concept of edge pivoting which allows us to collect 2-hop information without maintaining an explicit 2-hop neighborhood list at each vertex. This enables the computation of all the local 3-profiles in parallel with minimal communication.

We test our implementation in several experiments scaling up to 640 cores on Amazon EC2. We find that our algorithm can estimate the 3-profile of a graph in approximately the same time as triangle counting. For the harder problem of ego 3-profiles, we introduce an algorithm that can estimate profiles of hundreds of thousands of vertices in parallel, in the timescale of minutes.

1. INTRODUCTION
Given a small integer \( k \) (e.g. \( k = 3 \) or 4), the \( k \)-profile of a graph \( G(V, E) \) is a vector with one coordinate for each distinct \( k \)-node graph \( H_i \) (see Figure 1 for \( k = 3 \)). Each coordinate counts the number of 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.

One can see \( k \)-profiles as a generalization of triangle (as well as other motif) counting problems. They are increasingly popular for graph analytics both for practical and theoretical reasons. They form a concise graph description that has found several applications for the web \([4, 25]\), social networks \([39]\), and biological networks \([27]\) and seem to be empirically useful. Theoretically, they connect to the emerging theory of graph homomorphisms, graph limits and graphons \([3, 39, 21]\).

In this paper we introduce a novel distributed algorithm for estimating the \( k = 3 \)-profiles of massive graphs. In addition to estimating the (global) 3-profile, we address two more general problems. One is calculating the local 3-profile for each vertex \( v_j \). This assigns a vector to each vertex that counts how many times \( v_j \) participates in each subgraph \( H_i \). These local vectors contain a higher resolution description of the graph and are used to obtain the global 3-profile (simply by rescaled addition as we will discuss).

The second related problem is that of calculating the ego 3-profile for each vertex \( v_j \). This is the 3-profile of the graph \( N(v_j) \) i.e. the neighbors of \( v_j \), also called the ego graph of \( v_j \). The 3-profile of the ego graph of \( v_j \) can be seen as a projection of the vertex into a coordinate system \([30]\). This is a very interesting idea of viewing a big graph as a collection of small dense graphs, in this case the ego graphs of the vertices. Note that calculating the ego 3-profiles for a set of vertices of a graph is different (in fact, significantly harder) than calculating local 3-profiles.

Contributions: Our first contribution is a provable edge sub-sampling scheme: we establish sharp concentration results for estimating the entire 3-profile of a graph. This allows us to randomly discard most edges of the graph and still have 3-profile estimates that are provably within a bounded...
error with high probability. Our analysis is based on modeling the transformation from original to sampled graph as a one step Markov chain with transitions expressed as a function of the sampling probability. Our result is that a random sampling of edges forms a 3-profile sparsifier, i.e., a subgraph that preserves the elements of the 3-profile with sufficient probability concentration. Our result is a generalization of the Triangle sparsifiers by Tsourakakis et al. [38]. Our proof relies on a result by Kim and Vu [15] on concentration of multivariate polynomials, similarly to [38]. Unfortunately, the Kim and Vu concentration holds only for a class of polynomials called totally positive and some terms in the 3-profile do not satisfy this condition. For that reason, the proof of [38] does not directly extend beyond triangles. Our technical innovation involves showing that it is still possible to decompose our polynomials as combinations of totally positive polynomials using a sequence of variable changes.

Our second innovation deals with efficiently designing a distributed algorithm for estimating 3-profiles on the sub-sampled graph. We rely on the Gather-Apply-Scatter model used in GraphLab Powergraph [9] but, more generally, our algorithm fits the architecture of most graph engines. We introduce the concept of edge pivoting which allows us to collect 2-hop information without maintaining an explicit 2-hop neighborhood list at each vertex. This enables the computation of all the local 3-profiles in parallel. Each edge requires only information from its endpoints and each vertex only computes quantities using data from incident edges. For the problem of ego 3-profiles, we show how to combine edge pivot equations and local clique counts.

We implemented our algorithm in GraphLab and performed several experiments scaling up to 640 cores on Amazon EC2. We find that our algorithm can accurately estimate the 3-profile of a graph in approximately the same time as triangle counting. Specifically, we compare against the Powergraph triangle counting routine and find that it takes us only 1%-10% more time to compute the full 3-profile. For the significantly harder problem of ego 3-profiles, we were able to compute (in parallel) the 3-profiles of up to 100,000 ego graphs in the timescale of several minutes. We compare our parallel ego k-profile algorithm to a simple sequential algorithm that operates on each ego graph sequentially and show tremendous scalability benefits, as expected. Our datasets involve social network and web graphs with edges ranging in number from tens of millions to over one billion. We present results on both overall runtimes and network communication on multicore and distributed systems.

2. RELATED WORK

In this section, we describe several related topics and discuss differences in relation to our work.

Graph Sub-Sampling: Random edge sub-sampling is a natural way to quickly obtain estimates for graph parameters. For the case of triangle counting such graphs are called triangle sparsifiers [38]. Related ideas were explored in the DOULION algorithm [36, 37, 38] with increasingly strong concentration bounds. The recent work by Ahmed et al. [1] develops subgraph estimators for clustering coefficient, triangle count, and wedge count in a streaming sub-sampled graph. Other recent work [22, 13, 37, 33, 3] uses random sampling to estimate parts of the 3 and 4-profile. These methods do not account for a distributed computation model and require more complex sampling rules. As discussed, our theoretical results build on [38] to define the first 3-profile sparsifiers, sparse graphs that are a fortiori triangle sparsifiers.

Triangle Counting in Graph Engines: Graph engines (e.g., Pregel, GraphLab, Galois, GraphX, see [30] for a comparison) are frameworks for expressing distributed computation on graphs in the language of vertex programs. Triangle counting algorithms [31, 4] form one of the standard graph analytics tasks for such frameworks [9, 30]. In [7], the authors list triangles efficiently, by partitioning the graph into components and processing each component in parallel. Typically, it is much harder to perform graph analytics over the MapReduce framework but some recent work [26, 35] has used clever partitioning and provided theoretical guarantees for triangle counting.

Matrix formulations: Fast matrix multiplication has been used for certain types of subgraph counting. Alon et al. proposed a cycle counting algorithm which uses the trace of a matrix power on high degree vertices [2]. Some of our edge pivot equations have appeared in [16, 17, 20], all in a centralized setting. Related approximation schemes [36] and randomized algorithms [10] depend on centralized architectures and computing matrix powers of very large matrices.

Frequent Subgraph Discovery: The general problem of finding frequent subgraphs, also known as motifs or subgraph isomorphisms, is to find the number of occurrences of a small query graph within a larger graph. Typically frequent subgraph discovery algorithms offer pruning rules to eliminate false positives early in the search [41, 19, 10, 28, 29]. This is most applicable when subgraphs have labeled vertices or directed edges. For these problems, the number of unique isomorphisms grows much larger than in our application.

In [39], subgraphs were queried on the ego graphs of users. While enumerating all 3-sets and sampling 4-sets neighbors can be done in parallel, forming the ego subgraphs requires checking for edges between neighbors. This suggests that a graph engine implementation would be highly preferable over an Apache Hive system. Our algorithms simultaneously compute the ego subgraphs and their profiles, reducing the amount of communication between nodes. Our algorithm is suitable for both NUMA multicore and distributed architectures, but our implementation focus in this paper is on GraphLab.

Graphlets: First described in [27], graphlets generalize the concept of vertex degree to include the connected subgraphs a particular vertex participates in with its neighbors. Unique graphlets are defined at a vertex based on its degree. Graphlet frequency distributions (GFDs) have proven extremely useful in the field of bioinformatics. Specifically, GFD analysis of protein interaction networks helps to design improved generative models [11], accurate similarity measures [27], and better features for classification [21, 34]. Systems that use our edge pivot equations (in a different form) appear in prior literature for calculating GFDs [12, 22] but not for enabling distributed computation.

3. UNBIASED 3-PROFILE ESTIMATION

In this section, we are interested in estimating the number of 3 node subgraphs of type $H_0$, $H_1$, $H_2$ and $H_3$, as depicted in Figure 1 in a given graph $G$. Let the estimated counts be denoted $X_i$, $i \in \{0, 1, 2, 3\}$. Let the actual counts be...
In the graph, denoted with the following vector:

$$\mathbf{n}_3(G) = [n_0, n_1, n_2, n_3].$$  \hspace{1cm} (1)

Because the vector is a scaled probability distribution, there are only 3 degrees of freedom. Therefore, we calculate

$$\mathbf{n}_3(G) = \left[ \left( \frac{|V|}{3} \right) - n_1 - n_2 - n_3, n_1, n_2, n_3 \right].$$

In the case of a large graph, computational difficulty in estimating the 3-profile depends on the total number of edges in the large graph. We would like to estimate each of the 3-profile counts within a multiplicative factor. So we first sub-sample the set of edges in the graph with probability \( p \). We compute all 3-profile counts of the sub-sampled graph exactly. Let \( Y_i \) denote the exact 3-profile counts of the sub-sampled graph. We relate the sub-sampled 3-profile counts to the original ones through a one step Markov chain involving transition probabilities. The sub-sampling process is the random step in the chain. Any specific sub-graph is preserved with some probability and otherwise transitions to one of the other subgraphs. For example, a 3-clique is preserved with probability \( p^3 \). Figure 2 illustrates the other transition probabilities.

In expectation, this yields the following system of equations:

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

from which we obtain unbiased estimators for each entry in \( \mathbf{X}(G) = [X_0, X_1, X_2, X_3] \):

$$X_0 = Y_0 - \frac{1 - p}{p} Y_1 + \frac{(1 - p)^2}{p^2} Y_2 - \frac{(1 - p)^3}{p^3} Y_3, \hspace{1cm} (3)$$

$$X_1 = \frac{1}{p} Y_1 - \frac{2(1 - p)}{p^2} Y_2 + \frac{3(1 - p)^2}{p^3} Y_3, \hspace{1cm} (4)$$

$$X_2 = \frac{1}{p} Y_2 - \frac{3(1 - p)}{p^3} Y_3, \hspace{1cm} (5)$$

$$X_3 = \frac{1}{p^3} Y_3. \hspace{1cm} (6)$$

**Lemma 1.** \( \mathbf{X}(G) \) is an unbiased estimator of \( \mathbf{n}(G) \).

**Proof.** By substituting (3)–(6) into (2), clearly \( E[X_i] = n_i \) for \( i = 0, 1, 2, 3 \). \( \Box \)

We now turn to prove concentration bounds for the above estimators. We introduce some notation for this purpose. Let \( X \) be a real polynomial function of \( m \) real random variables \( \{t_i\}_{i=1}^m \). Let \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m) \in \mathbb{Z}_+^m \) and define

$$E_{\geq 1}[X] = \max_{\alpha} E (\partial^\alpha X),$$

where

$$E(\partial^\alpha X) = E \left[ (\frac{\partial}{\partial t_1})^{\alpha_1} \ldots (\frac{\partial}{\partial t_m})^{\alpha_m} [X(t_1, \ldots, t_m)] \right].$$  \hspace{1cm} (7)

Further, we call a polynomial totally positive if the coefficients of all the monomials involved are non-negative. We state the main technical tool we use to obtain our concentration results.

**Theorem 1 (Kim-Vu Concentration [15]).** Let \( X \) be a random totally positive Boolean polynomial in \( m \) boolean random variables with degree at most \( k \). If \( E[X] \geq E_{\geq 1}[X] \), then

$$P \left[ X - E[X] > \alpha_k \sqrt{E[X]E_{\geq 1}[X] \lambda^k} \right] = O \left( \exp (-\lambda + (k - 1) \log m) \right)$$  \hspace{1cm} (8)

for any \( \lambda > 1 \), where \( \alpha_k = \lambda^k k^{1/2} \).

The above theorem was used to analyze 3-profiles of Erdős-Rényi random ensembles \((G_n,p)\) in [15]. Later, this was used to derive concentration bounds for triangle sparsifiers in [38]. Here, we extend §4.3 of [15] for concentration bounds for the 3-profile estimation process, detailed above, on an arbitrary edge-sampled graph.

**Theorem 2.** (Generalization of Triangle sparsifiers to 3-profile sparsifiers) Let \( \mathbf{n}(G) = [n_0, n_1, n_2, n_3] \) be the 3-profile of a graph \( G(V,E) \). Let \( |V| = n \) and \( |E| = m \). Let \( \mathbf{n}(G) = [Y_0, Y_1, Y_2, Y_3] \) be the 3-profile of the subgraph obtained by sampling each edge in \( G \) with probability \( p \). Let \( \alpha, \beta \) and \( \Delta \) be the largest collection of \( H_1 \)'s, wedges and triangles that share a common edge. Define \( \mathbf{X}(G) \) according to (3)–(6), \( \epsilon > 0 \), and \( \gamma > 0 \). If \( p \), \( \epsilon \) satisfy:

$$\frac{n_0}{3 \max\{\alpha, \beta, \Delta\}} \geq \frac{a_0^2 \log^6 (m^{2+\gamma})}{\epsilon^2}$$

$$\frac{p}{\max\{\frac{\alpha}{\alpha_n}, \frac{\beta}{\beta_n}, \frac{\Delta}{\Delta_n}\}} \geq \frac{a_0^2 \log^6 (m^{2+\gamma})}{\epsilon^2}$$

$$\frac{p}{\alpha/\alpha_n} \geq \frac{a_1^2 \log^2 (m^{\gamma})}{\epsilon^2}$$

$$\frac{p}{\max\{\frac{\beta}{\beta_n}, \frac{\Delta}{\Delta_n}\}} \geq \frac{a_2^2 \log^4 (m^{1+\gamma})}{\epsilon^2},$$  \hspace{1cm} (9)

Figure 2: Edge sampling process
is easy to see that the global counts can be obtained from final 4 scalars (first outline an approach that calculates these counts and note the corresponding local subgraph counts at $v_1$). 

The sampling probability $p$ in Theorem 2 depends polynomially on the number of edges and linearly on the fraction of each subgraph which occurs on a common edge. For example, if all of the wedges in $G$ depend on a single edge, i.e. $\beta = n_2$, then the last equation suggests the presence of that particular edge in the sampled graph will dominate the overall sparsity quality.

4. LOCAL 3-PROFILE CALCULATION

In this section, we describe how to obtain two types of 3-profiles for a given graph $G$ in a deterministic manner. These algorithms are distributed and can be applied independently of the edge sampling described in Section 3.

The key to our approach is to identify subgraphs at a vertex based on degree with which it participates in the subgraph. From the perspective of a given vertex $v$, there are actually six distinct 3 node subgraphs up to isomorphism as given in Figure 3. Let $n_{0,v}, n_{1,v}, n_{2,v}, n_{2,v}^e, n_{3,v}, n_{3,v}^e$ denote the corresponding local subgraph counts at $v$. We will first outline an approach that calculates these counts and then add across different vertex perspectives to calculate the final 4 scalars ($n_{2,v} = n_{2,v}^e + n_{2,v}$ and $n_{1,v} = n_{1,v} + n_{1,v}^e$). It is easy to see that the global counts can be obtained from these local counts by summing across vertices:

$$n_i = \frac{1}{3} \left( \sum_{v \in V} n_{i,v} \right), \forall i.$$ (10)

4.1 Distributed Local 3-profile

We will now give our approach for calculating the local 3-profile counts of $G(V, E)$ using only local information combined with $|V|$ and $|E|$.

**Scatter:** We assume that every edge $(v, a)$ has access to the neighborhood sets of both $v$ and $a$, i.e. $\Gamma(v)$ and $\Gamma(a)$. Therefore, intersection sizes are first calculated at every edge, i.e. $|\Gamma(v) \cap \Gamma(a)|$. Each edge computes the following scalars and stores them:

$$n_{3,v,a} = |\Gamma(v) \cap \Gamma(a)|, n_{2,v,a} = |\Gamma(v)| - |\Gamma(v) \cap \Gamma(a)| - 1,$$

$$n_{2,v,a} = |\Gamma(a)| - |\Gamma(v) \cap \Gamma(a)| - 1,$$

$$n_{1,v,a} = |V| - (|\Gamma(v)| + |\Gamma(a)| - |\Gamma(v) \cap \Gamma(a)|).$$ (11)

The computational effort at every edge is at most $O(d_{\text{max}})$, where $d_{\text{max}}$ is the maximum degree of the graph, for the neighborhood intersection size.

4.2 Distributed Ego 3-profile

In this section, we give an approach to compute ego 3-profiles for a set of vertices $V \subseteq G$. For each vertex $v$, the algorithm returns a 3-profile corresponding to that vertex’s ego $N(v)$, a subgraph induced by the neighborhood set $\Gamma(v)$, including edges between neighbors and excluding $v$ itself. Formally, our goal is to compute $N(N(v))$ for $v \in V$. Clearly, this can be accomplished in two steps repeated serially on all $v \in V$: first obtain the ego subgraph $N(v)$ and then pass as input to Algorithm 1 summing over the ego vertices $\Gamma(v)$ to get a global count. The serial implementation is provided in Algorithm 2. We note that this was essentially done in [39], where ego subgraphs were extracted from a common graph separately from 3-profile computations.

**Scatter:** We assume that every edge $(v, a)$ has already computed the scalars from (11). Additionally, every edge $(v, a)$ also computes the list $N_{v,a} = \Gamma(v) \cap \Gamma(a)$ instead of only its size. The computational complexity is still $O(d_{\text{max}})$.

Gather: In the next round, vertex $v$ “gathers” the above scalars in the following way:

$$n_{2,v} = \sum_{a \in \Gamma(v)} n_{2,v,a}, \quad n_{1,v} = \sum_{a \in \Gamma(v)} n_{1,v,a},$$

$$n_{2,v}^e = \frac{1}{2} \sum_{a \in \Gamma(v)} n_{2,v,a}, \quad n_{3,v}^e = \frac{1}{2} \sum_{a \in \Gamma(v)} n_{3,v,a},$$

$$n_{1,v}^e = |E| - |\Gamma(v)| - n_{3,v} - n_{2,v}^e,$$

$$n_{0,v} = \left( |V| - 1 \right) - n_{1,v} - n_{2,v} - n_{3,v}.$$ (12)

Here, relations (a) and (b) are because triangles and wedges from center are double counted. (c) comes from noticing that each triangle and wedge from endpoint excludes an extra edge from forming $H_3^e(v)$. In this Gather stage, the communication complexity is $O(M)$ where it is assumed that $\Gamma(v)$ is stored over $M$ different machines. The corresponding distributed algorithm is described in Algorithm 4.
By choosing two subgraphs that the edge \((v,a)\) participates in, and then summing over neighbors \(a\), these equations gather implicit connectivity information 2 hops away from \(v\). However, note that there are only three equations in four variables and we must count one of them directly, namely the number of 4-cliques \(F_3(v)\). Therefore, at the same gather step, the vertex also creates the list \(\mathcal{CN}_v = \bigcup_{a \in \Gamma(v)} \{a, p\}\). Essentially, this is the list of edges in the subgraph induced by \(\Gamma(v)\). This requires worst case communication proportional to the number of edges in \(N(v)\), independent of the number of machines \(M\).

**Scatter:** Now, at the next scatter stage, each edge \((v,a)\) receives the pair of lists \(\mathcal{CN}_v, \mathcal{CN}_a\). This again incurs communication of \(O(|\mathcal{CN}(v)|) = O(M d_{max})\). Now, each edge \((v,a)\) computes the number of 4-cliques it is a part of, defined as follows:

\[
\sum_{i,j \in \Gamma(v) \cap \Gamma(a)} 1 \left( (i, j) \in \mathcal{CN}_v \right). \tag{14}
\]

This incurs computation time of \(|\mathcal{CN}_v|\).

**Gather:** In the final gather stage, every vertex \(v\) accumulates these scalars to get \(F_3(v) = \frac{1}{3} \sum_{a \in \Gamma(v)} n_{4,v,a}\) requiring \(O(M)\) communication time. As in the previous section, the scaling accounts for extra counting. Finally, the vertex solves the equations (13) using \(F_3(v)\).

## 5. IMPLEMENTATION AND RESULTS

In this section, we describe the implementation and the experimental results of the 3-PROF, EGO-PAR and EGO-SER algorithms. We implement our algorithms on GraphLab v2.2 (PowerGraph) [9]. The performance (running time and network usage) of our 3-PROF algorithm is compared with the Undirected Triangles Count per Vertex (hereinafter referred to as TRIAN) algorithm shipped with GraphLab. We show that in time and network usage comparable to the built-in TRIAN algorithm, our 3-PROF can calculate all the local and global 3-profiles. Then, we compare our parallel implementation of the ego 3-profile algorithm, EGO-PAR, with the naive serial implementation, EGO-SER. It appears that our parallel approach is much more efficient and scales much better than the serial algorithm. The sampling approach, introduced for the 3-PROF algorithm, yields promising results – reduced running time and network usage while still providing excellent accuracy. We support our findings with several experiments over various data sets and systems.

**Vertex Programs:** Our algorithms are implemented using a standard GAS (gather, apply, scatter) model [4]. We implement the three functions gather( ), apply( ), and scatter( ) to be executed by each vertex. Then we signal subsets of vertices to run in a specific order.

### Algorithm 1 3-PROF

**Input:** Graph \(G(V,E)\) with \(|V|\) vertices, \(|E|\) edges

**Gather:** For each vertex \(v\) union over edges of the ‘other’ vertex in the edge, \(\cup_{a \in \Gamma(v)} a = \Gamma(v)\)

**Apply:** Store the gather as vertex data \(v_nb\), size automatically stored

**Scatter:** For each edge \(e_{va}\), compute and store scalars in \(g\).

**Gather:** For each vertex \(v\), sum edge scalar data of neighbors

\(g \leftarrow \sum_{e_{va} \in \Gamma(v)} e.data\).

**Apply:** For each vertex \(v\), calculate and store the quantities described in (12).

**return** \([v: v.n0 v.n1 v.n2 v.n3]\)

### Algorithm 2 EGO-SER

**Input:** Graph \(G(V,E)\) with \(|V|\) vertices, \(|E|\) edges, set of ego vertices \(V\)

**for** \(v \in V\) **do**

Signal \(v\) and its neighbors

Include an edge if both its endpoints are signaled

Run Algorithm 1 on the graph induced by the neighbors and edges between them

**end for**

**return** \([v: vego.n0 vego.n1 vego.n2 vego.n3]\)

### The Systems:

We perform the experiments on three systems. The first system is a single power 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 next two systems are EC2 clusters on AWS (Amazon Web Services) [3]. One is comprised of 12 m3.2xlarge machines, each having 30 GB RAM and 8 virtual CPUs. Another system is a cluster of 20 c3.8xlarge machines, each having 60 GB RAM and 32 virtual CPUs.

### The Data:

In our experiments we used five real graphs. These graphs represent different datasets: social networks (LiveJournal and Twitter), citations (DBLP), knowledge content (Wikipedia), and WWW structure (PLD – pay level domains). Graph sizes are summarized in Table 1.

### Table 1: Datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>Vertices</th>
<th>Edges (undirected)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twitter</td>
<td>18</td>
<td>41,652,230</td>
</tr>
<tr>
<td>PLD</td>
<td>23</td>
<td>39,497,204</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>20</td>
<td>4,846,609</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>8</td>
<td>3,515,067</td>
</tr>
<tr>
<td>DBLP</td>
<td>20</td>
<td>317,080</td>
</tr>
</tbody>
</table>

### 5.1 Results

**Graph 3-profile vs triangle count:** The first result is that our 3-PROF is able to compute all the local 3-profiles in almost the same time as the GraphLab’s built-in TRIAN computes the local triangles (i.e., number of triangles including each vertex). Let us start with the first AWS cluster with less powerful machines (m3.xlarge). In Figure 5 (a)
Finally, we show that while the sampling approach can significantly reduce the running time and network usage, it has negligible effect on the accuracy of the solution. Notice that the sampling accuracy refers to the global 3-profile count (i.e., the sum of all the local 3-profiles over all vertices in a graph). In Figure 12 we show accuracy of each of the four 3-profiles. For the accuracy metrics, we use ratio between the exact count (obtained running 3-PROF with $p = 1$) divided by the estimated count (i.e., the output of our 3-PROF when $p < 1$). It can be seen that for the three graphs, all the 3-profiles are very close to 1. E.g., for the PLD graph, even when $p = 0.01$, the accuracy is within 0.004 from the ideal value of 1. Error bars mark one standard deviation from the mean, and across all graphs the largest standard deviation is 0.031. As $p$ decreases, the triangle estimator suffers the greatest loss in both accuracy and consistency.

![Figure 5: AWS m3.2xlarge cluster. 3-prof vs trian algorithms for LiveJournal and Wikipedia datasets (average of 3 runs). 3-prof achieves comparable performance to triangle counting. (a,b) – Running time for various numbers of nodes (machines) and various sampling probabilities $p$. (c,d) – Network bytes sent by the algorithms for various numbers of nodes and various sampling probabilities $p$.](image)

Ego 3-profiles: The next set of experiments evaluates the performance of our EGO-PAR algorithm for counting ego 3-profiles. We show the performance of EGO-PAR for various graphs and systems and also compare it to a naive serial algorithm EGO-SER. Let us start with the AWS system with $(c3.8xlarge$ machines). In Figure 5 we see the running time of EGO-SER and EGO-PAR on the LiveJournal graph. The task was to find ego 3-profiles of 100, 1K, and 10K randomly selected nodes. Since the running time depends on the size and structure of each induced subgraph, EGO-SER and EGO-PAR operated on the same list of ego vertices. While for 100 random vertices EGO-SER performed well (and even achieved the same running time as EGO-PAR for the PLD graph), its performance drastically degraded for a larger number of vertices. This is due to its itera-
tive nature – it finds ego 3-profiles of the vertices one at a time and is not scalable. Note that the open bars, and the numbers above them, mean that this experiment was not finished and the number is approximate interpolation, which is reasonable due to the serial design of the Ego-ser.

On the contrary, Ego-par algorithm scales extremely well and computes ego 3-profiles of 100, 1K, and 10K vertices almost in the same time. In Figure 6(a), we can see that as the number of nodes (i.e., machines) increases, running time of Ego-par decreases since its parallel design allows it to use additional computational resources. However, the Ego-ser cannot benefit from more resources and its running time increases when more machines are used. The increase in running time of Ego-ser is due to the increase in network usage when using more machines (see Figure 6(b)). The network usage of Ego-par also increases, but it is compensated with the additional computational power that it can leverage. In Figure 6(c) we can see that Ego-par performed well even when finding ego 3-profiles for all the LiveJournal vertices (4.8M vertices).

Finally in Figure 6(d) and (e), we can see the comparison of Ego-par and Ego-ser on the PLD and the DBLP graphs on the Asterix machine. For both graphs, we can see a very good scaling of Ego-par, while the running time of Ego-ser scales linearly with the size of the ego vertices list.

6. CONCLUSIONS

In summary, we have reduced several 3-profile problems to triangle and 4-clique finding in a graph engine framework. Our concentration theorem and experimental results confirm that local 3-profile estimation via sub-sampling is comparable in runtime and accuracy to local triangle counting.

This paper offers several directions for future work. First, both the local 3-profile and the ego 3-profile can be used as features to classify vertices in social or bioinformatic networks. Additionally, we hope to extend our theory and algorithmic framework to larger subgraphs, as well as special classes of input graphs. Our edge sampling Markov chain and unbiased estimators should easily extend to $k > 3$. Equations in (13) are useful to count local or global 4-profiles in a centralized setting, as shown recently in [17, 40]. Tractable distributed algorithms for $k > 3$ using similar edge pivot equations remain as future work. Our observed dependence on 4-clique count suggests that an improved graph engine-based clique counting subroutine will improve the parallel algorithm’s performance.

7. REFERENCES


Ego-ser 16 nodes
Full graph
LiveJournal, AWS c3_8xlarge 20 nodes
(c)
PLD
(b)
Ego-par
1K egos
Ego-par 20 nodes
1K egos
Ego-ser 20 nodes
10K egos
(b)
(b)
10K egos
10K egos
1K egos
Ego-par 16 nodes

Running time [sec]
Running time [sec]
Running time [sec]
Network sent [bytes]
Network sent [bytes]
Network sent [bytes]

Running time [sec]
Running time [sec]
Running time [sec]

Running time [sec]
Running time [sec]
Running time [sec]

Running time [sec]
Running time [sec]
Running time [sec]

Running time [sec]
Running time [sec]
Running time [sec]

Running time [sec]
Running time [sec]
Running time [sec]

Running time [sec]
Running time [sec]
Running time [sec]

Ego-ser 12 nodes
Ego-ser 16 nodes
Ego-ser 20 nodes
Ego-par 12 nodes
Ego-par 16 nodes
Ego-par 20 nodes

Figure 10: AWS c3_8xlarge cluster. Ego-par vs Ego-ser results for LiveJournal and Wikipedia datasets (average of 5 runs). Running time of Ego-par decreases with the number of machines due to its parallel design. Running time of Ego-ser does not decrease with the number of machines due to its iterative nature. Network usage increases for both algorithms with the number of machines.

Twitter and PLD, Asterix
PLD, Asterix
DBLP, Asterix

Figure 11: Asterix machine. Results for Twitter, PLD, and DBLP datasets. (a) - Running time of 3-prof vs trian for various sampling probabilities p. (b,c) - Running time of Ego-par vs Ego-ser for various number of ego centers. Results are averaged over 3, and 3, and 10 runs, respectively.

Figure 10: AWS c3_8xlarge cluster with 20 nodes. Ego-par results for LiveJournal dataset (average of 5 runs). The algorithm scales well for various number of ego centers and even full ego centers list. (a) - Running time. (b) - Network bytes sent by the algorithm.

Figure 12: Global 3-profiles accuracy achieved by 3-prof algorithm for various graphs and for each profile count. Results are averaged over 5, and 5, and 10 iterations, respectively. Error bars indicate 1 standard deviation. The metric is a ratio between the exact profile count (when \( p = 1 \)) and the given output for \( p < 1 \). All results are very close to the optimum value of 1.

[36] C. E. Tsourakakis. Fast Counting of Triangles in
APPENDIX

A. PROOF OF THEOREM 2

Let $m$ be the total number of edges in the original graph $G$. If $e$ is an edge in the original graph $G$, let $t_e$ be the random indicator after sampling. $t_e = 1$ if $e$ is sampled and 0 otherwise. Let $H_0, H_1, H_2, H_3$ denote the set of distinct subgraphs of the kind $H_0, H_1, H_2$ and $H_3$ (anti-clique, edge, wedge and triangle) respectively. Let $\Lambda(e), \Lambda(e, f)$ and $\Delta(e, f, g)$ denote an anti-clique with no edge, a $H_1$ with edge $e$, a $H_2$ with two edges $e, f$ and a triangle with edges $e, f, g$ respectively in the original graph $G$. Our estimators $S_1, S_2, D_1, D_2$ are a function of $Y_i$’s and each $Y_i$ can be written as a polynomial of at most degree 3 in all the variables $t_e$.

\[
Y_0 = n_0 + \sum_{\Lambda(e) \in H_1} (1 - t_e) + \sum_{\Lambda(e, f) \in H_2} (1 - t_e)(1 - t_f) + \sum_{\Delta(e, f, g) \in H_3} (1 - t_e)(1 - t_f)(1 - t_g)
\]

(15)

\[
Y_1 = \sum_{\Lambda(e) \in H_1} t_e + \sum_{\Lambda(e, f) \in H_2} ((1 - t_e)t_f + (1 - t_f)t_e) + \sum_{\Delta(e, f, g) \in H_3} t_e(1 - t_f)(1 - t_g) + \sum_{\Delta(e, f, g) \in H_3} t_f(1 - t_e)(1 - t_g) + t_g(1 - t_e)(1 - t_f)
\]

(16)

\[
Y_2 = \sum_{\Lambda(e, f) \in H_2} t_et_f + \sum_{\Delta(e, f, g) \in H_3} (t_et_f)(1 - t_g) + t_ftp_g(1 - t_e) + t_et_f(1 - t_f)t_g
\]

(17)

\[
Y_3 = \sum_{\Delta(e, f, g) \in H_3} t_et_ft_g
\]

(18)

\[
S_1 = \sum_{\Lambda(e) \in H_1} t_e
\]

(19)

\[
D_1 = \sum_{\Lambda(e, f) \in H_2} (t_e + t_f)
\]

(20)

\[
D_2 = \sum_{\Delta(e, f, g) \in H_3} t_et_f
\]

(21)

\[
T_1 = \sum_{\Delta(e, f, g) \in H_3} (t_e + t_f + t_g)
\]

(22)

\[
T_2 = \sum_{\Delta(e, f, g) \in H_3} (t_et_f + t_f t_g + t_g t_e)
\]

(23)

\[
Y_1 = S_1 + D_1 - 2D_2 + T_1 - 2T_2 + 3Y_3
\]

(24)

\[
Y_2 = D_2 + T_2 - 3Y_3
\]

(25)

Note that the newly defined polynomials have the following expectations:

\[
E[S_1] = pm_1
\]

\[
E[D_1] = 2pm_2
\]

\[
E[D_2] = p^2n_2
\]

\[
E[T_1] = 3pn_3
\]

\[
E[T_2] = 3p^2n_3.
\]

We observe that in the above even by change of variables $y_e = (1 - t_e), Y_1$ and $Y_2$ are not totally positive polynomials. This means that Theorem 2 cannot be applied directly to the $Y_i$’s or $X_i$’s. The strategy we adopt is to split the $Y_i$ and $Y_2$ into many polynomials, each of which is totally positive, and then apply Theorem 2 on each of them. $P = \{Y_0, Y_3, S_1, D_1, D_2, T_1, T_2\}$ form the set of totally positive polynomials (proved below). Substituting the above equations into (3)-(6), we have the following system of equations that connect $X_i$’s and the set of totally positive poly-
nominals $P$:

$$X_0 = Y_0 - \frac{1-p}{p} (S_1 + D_1 + T_1 - 2D_2 - 2T_2 + 3Y_3)$$

$$+ \frac{(1-p)^2}{p^2} (D_2 + T_2 - 3Y_3) - \frac{(1-p)^3}{p^3} Y_3$$

$$= Y_0 - \frac{1-p}{p} (S_1 + D_1 + T_1)$$

$$+ \frac{1-p^2}{p^2} (D_2 + T_2) - \frac{1-p^3}{p^3} Y_3. \quad (26)$$

$$X_1 = \frac{1}{p} (S_1 + D_1 + T_1 - 2D_2 - 2T_2 + 3Y_3)$$

$$- \frac{2(1-p)}{p^2} (D_2 + T_2 - 3Y_3) + \frac{3(1-p)^2}{p^3} Y_3$$

$$= \frac{1}{p} (S_1 + D_1 + T_1) - \frac{2}{p^2} (D_2 + T_2) + \frac{3}{p^3} Y_3. \quad (27)$$

$$X_2 = \frac{1}{p^2} (D_2 + T_2 - 3Y_3) - \frac{3(1-p)}{p^3} Y_3$$

$$= \frac{1}{p^2} (D_2 + T_2) - \frac{3}{p^3} Y_3. \quad (28)$$

$$X_3 = \frac{1}{p^3} Y_3. \quad (29)$$

Let $\alpha$, $\beta$, and $\Delta$, be the maximum number of $H_1$’s, $H_2$’s, and $H_3$’s containing an edge $e$ in the original graph $G$. Let $\alpha$, $\beta$ and $\Delta$ be the maximum of $\alpha$, $\beta$, and $\Delta$ over all edges $e$.

We now show concentration results for the totally positive polynomials alone.

**Lemma 1.** Define variables $y_e = 1 - t_e$. Then $Y_0$ is totally positive in $y_e$. With respect to the variables $y_e$, $\mathbb{E}_{\geq 1} \{Y_0\} \leq 3 \max \{\alpha, \beta, \Delta\}$

**Proof.** We have the expectation of the following partial derivatives, up to the third order:

$$\mathbb{E} \left[ \frac{\partial Y_0}{\partial y_e} \right] = \alpha_e + (1-p)\beta_e + (1-p)^2 \Delta_e$$

$$\leq 3 \max \{\alpha, \beta, \Delta\}. \quad (30)$$

To satisfy $\mathbb{E}_{\geq 1} \{Y_0\} \leq \mathbb{E} \{Y_0\}$, it is sufficient to have

$$n_0 \geq 3 \max \{\alpha, \beta, \Delta\}. \quad (31)$$

This is because $Y_0 \geq n_0$ with probability 1.

**Lemma 2.** $Y_3$ is totally positive in $t_e$. With respect to the variables $t_e$, $\mathbb{E}_{\geq 1} \{Y_3\} \leq \max \{1, p^2 \Delta\}$

**Proof.** We have the expectation of the following partial derivatives, up to the third order:

$$\mathbb{E} \left[ \frac{\partial Y_3}{\partial t_e} \right] = p^2 \Delta_e, \mathbb{E} \left[ \frac{\partial Y_3}{\partial t_e t_f} \right] = p, \mathbb{E} \left[ \frac{\partial Y_3}{\partial t_e t_f t_g} \right] \leq 1.$$

From the above equations, we have $\mathbb{E}_{\geq 1} \{Y_3\} \leq 3 \max \{1, p^2 \Delta\}$. A

**Lemma 3.** $S_1$ is totally positive in $t_e$. With respect to the variables $t_e$, $\mathbb{E}_{\geq 1} \{S_1\} \leq \alpha$.

**Proof.** We have the expectation of the following partial derivatives, up to the second order:

$$\mathbb{E} \left[ \frac{\partial S_1}{\partial t_e} \right] = \alpha_e, \mathbb{E} \left[ \frac{\partial S_1}{\partial t_e t_f} \right] = 0.$$

From the above equations, we have $\mathbb{E}_{\geq 1} \{S_1\} \leq \mathbb{E} \{S_1\}$ implies

$$p \geq \alpha/n_1. \quad (32)$$

**Lemma 4.** $D_1$ is totally positive in $t_e$. With respect to the variables $t_e$, $\mathbb{E}_{\geq 1} \{D_1\} \leq \beta$.

**Proof.** We have the expectation of the following partial derivatives, up to the second order:

$$\mathbb{E} \left[ \frac{\partial D_1}{\partial t_e} \right] = \beta_e, \mathbb{E} \left[ \frac{\partial D_1}{\partial t_e t_f} \right] = 0.$$

From the above equations, we have $\mathbb{E}_{\geq 1} \{D_1\} \leq \mathbb{E} \{D_1\}$ implies

$$p \geq \beta/(2n_2). \quad (33)$$

**Lemma 5.** $T_1$ is totally positive in $t_e$. With respect to the variables $t_e$, $\mathbb{E}_{\geq 1} \{T_1\} \leq \Delta$.

**Proof.** We have the expectation of the following partial derivatives, up to the second order:

$$\mathbb{E} \left[ \frac{\partial T_1}{\partial t_e} \right] = \Delta_e, \mathbb{E} \left[ \frac{\partial T_1}{\partial t_e t_f} \right] = 0.$$

From the above equations, we have $\mathbb{E}_{\geq 1} \{T_1\} \leq \mathbb{E} \{T_1\}$ implies

$$p \geq \Delta/(3n_3). \quad (34)$$

**Lemma 6.** $D_2$ is totally positive in $t_e$. With respect to the variables $t_e$, $\mathbb{E}_{\geq 1} \{D_2\} \leq \max \{p^2 \Delta, 1\}$.

**Proof.** We have the expectation of the following partial derivatives, up to the second order:

$$\mathbb{E} \left[ \frac{\partial D_2}{\partial t_e} \right] = p \beta_e, \mathbb{E} \left[ \frac{\partial D_2}{\partial t_e t_f} \right] \leq 1.$$

From the above equations, we have $\mathbb{E}_{\geq 1} \{D_2\} \leq \mathbb{E} \{D_2\}$ implies

$$p \geq \max \{\beta/n_2, \frac{1}{\sqrt{n_2}}\}. \quad (35)$$

**Lemma 7.** $T_2$ is totally positive in $t_e$. With respect to the variables $t_e$, $\mathbb{E}_{\geq 1} \{T_2\} \leq \max \{2p \Delta, 1\}$.

**Proof.** We have the expectation of the following partial derivatives, up to the second order:

$$\mathbb{E} \left[ \frac{\partial T_2}{\partial t_e} \right] = 2p \Delta_e, \mathbb{E} \left[ \frac{\partial T_2}{\partial t_e t_f} \right] \leq 1.$$

From the above equations, we have $\mathbb{E}_{\geq 1} \{T_2\} \leq \max \{2p \Delta, 1\}$. □
$\mathbb{E}_2 [T_2] \leq \mathbb{E}[T_2]$ implies

$$p \geq \max \{ 2 \Delta / (3n_3), \frac{1}{\sqrt{3n_3}} \}.$$  \hfill (36)

Now merging all the conditions (30)–(36), we get

$$n_0 \geq 3 \max \{ \alpha, \beta, \Delta \}, \quad p \geq \max \{ \frac{1}{\sqrt{n_3}}, \frac{1}{\sqrt{n_2}}, \frac{\Delta}{n_3}, \frac{\beta}{n_2}, \frac{\alpha}{n_1} \}. \hfill (37)$$

Applying Theorem 2 to all the totally positive polynomials, along with condition 37, we get

$$P \left[ \left| Y_0 - \mathbb{E}[Y_0] \right| > a_3 \sqrt{\mathbb{E}[Y_0]\mathbb{E}_{\geq 1}[Y_0]} \lambda^3 \right] = \mathcal{O} (\exp (-\lambda_1 + (2) \log m))$$

$$P \left[ \left| Y_3 - \mathbb{E}[Y_3] \right| > a_3 \sqrt{\mathbb{E}[Y_3]\mathbb{E}_{\geq 1}[Y_3]} \lambda^3 \right] = \mathcal{O} (\exp (-\lambda_2 + (2) \log m))$$

$$P \left[ \left| S_1 - \mathbb{E}[S_1] \right| > a_1 \sqrt{\mathbb{E}[S_1]\mathbb{E}_{\geq 1}[S_1]} \lambda_3 \right] = \mathcal{O} (\exp (-\lambda_3))$$

$$P \left[ \left| D_1 - \mathbb{E}[D_1] \right| > a_1 \sqrt{\mathbb{E}[D_1]\mathbb{E}_{\geq 1}[D_1]} \lambda_4 \right] = \mathcal{O} (\exp (-\lambda_4))$$

$$P \left[ \left| T_1 - \mathbb{E}[T_1] \right| > a_1 \sqrt{\mathbb{E}[T_1]\mathbb{E}_{\geq 1}[T_1]} \lambda_5 \right] = \mathcal{O} (\exp (-\lambda_5))$$

$$P \left[ \left| D_2 - \mathbb{E}[D_2] \right| > a_2 \sqrt{\mathbb{E}[D_2]\mathbb{E}_{\geq 1}[D_2]} \lambda_6 \right] = \mathcal{O} (\exp (-\lambda_6 + \log m))$$

$$P \left[ \left| T_2 - \mathbb{E}[T_2] \right| > a_2 \sqrt{\mathbb{E}[T_2]\mathbb{E}_{\geq 1}[T_2]} \lambda_7 \right] = \mathcal{O} (\exp (-\lambda_7 + \log m)) \hfill (38)$$

Choose an $\epsilon > 0$. We force the following conditions:

$$a_3 \sqrt{\mathbb{E}[Y_0]\mathbb{E}_{\geq 1}[Y_0]} \lambda^3 = \epsilon \mathbb{E}[Y_0]$$

$$a_3 \sqrt{\mathbb{E}[Y_3]\mathbb{E}_{\geq 1}[Y_3]} \lambda^3 = \epsilon \mathbb{E}[Y_3]$$

$$a_1 \sqrt{\mathbb{E}[S_1]\mathbb{E}_{\geq 1}[S_1]} \lambda_3 = \epsilon \mathbb{E}[S_1]$$

$$a_1 \sqrt{\mathbb{E}[D_1]\mathbb{E}_{\geq 1}[D_1]} \lambda_4 = \epsilon \mathbb{E}[D_1]$$

$$a_1 \sqrt{\mathbb{E}[T_1]\mathbb{E}_{\geq 1}[T_1]} \lambda_5 = \epsilon \mathbb{E}[T_1]$$

$$a_2 \sqrt{\mathbb{E}[D_2]\mathbb{E}_{\geq 1}[D_2]} \lambda_6 = \epsilon \mathbb{E}[D_2]$$

$$a_2 \sqrt{\mathbb{E}[T_2]\mathbb{E}_{\geq 1}[T_2]} \lambda_7 = \epsilon \mathbb{E}[T_2]$$

Let $\gamma > 0$. For the right hand side of every equation in 38 to be $\mathcal{O}(\exp (-\gamma \log m))$, assuming all the bounds in

**Lemma 19**, it is sufficient to have

$$\frac{n_0}{3 \max \{ \alpha, \beta, \Delta \}} \geq \frac{a_3^2 \log^6 \left( \frac{m^{2+\gamma}}{\epsilon} \right)}{\epsilon^2}$$

$$\frac{p}{\max \{ \frac{\Delta}{n_3}, \frac{\beta}{n_2}, \frac{\alpha}{n_1} \}} \geq \frac{a_3^2 \log^6 \left( \frac{m^{2+\gamma}}{\epsilon} \right)}{\epsilon^2}$$

$$\frac{p}{\max \{ \frac{\alpha}{n_1}, \frac{\beta}{n_2}, \frac{\alpha}{n_1} \}} \geq \frac{a_3^2 \log^6 \left( \frac{m^{2+\gamma}}{\epsilon} \right)}{\epsilon^2}$$

$$\frac{p}{\max \{ \frac{\alpha}{n_1}, \frac{\beta}{n_2}, \frac{\alpha}{n_1} \}} \geq \frac{a_3^2 \log^6 \left( \frac{m^{2+\gamma}}{\epsilon} \right)}{\epsilon^2} \hfill (39)$$

We can see that conditions in 39 imply the conditions in 37. Further, conditions in 39 can be simplified to remove some redundancy as follows:

$$\frac{n_0}{3 \max \{ \alpha, \beta, \Delta \}} \geq \frac{a_3^2 \log^6 \left( \frac{m^{2+\gamma}}{\epsilon} \right)}{\epsilon^2}$$

$$\frac{p}{\max \{ \frac{\alpha}{n_1}, \frac{\beta}{n_2}, \frac{\alpha}{n_1} \}} \geq \frac{a_3^2 \log^6 \left( \frac{m^{2+\gamma}}{\epsilon} \right)}{\epsilon^2}$$

$$\frac{p}{\max \{ \frac{\alpha}{n_1}, \frac{\beta}{n_2}, \frac{\alpha}{n_1} \}} \geq \frac{a_3^2 \log^6 \left( \frac{m^{2+\gamma}}{\epsilon} \right)}{\epsilon^2} \hfill (40)$$

This is due to the fact that $a_3 \geq a_2 \geq a_1$ and $m^3 \geq m^2$. Therefore, subject to the conditions in 40, all totally positive polynomials $Y_0, Y_3, D_1, D_2, S_1, T_1, T_2$ concentrate within the multiplicative factor of $(1 + \epsilon)$ with probability at least $1 - \mathcal{O} \left( \frac{\epsilon}{m^\gamma} \right)$.

Under the above concentration result, let the deviations of $X_i$’s be denoted by $\delta X_i$. Now we calculate the deviation of $X_0$ using 26.

$$\delta X_0 \leq \epsilon \mathbb{E}[Y_0] + \frac{1 - p}{p} \left( |\mathbb{E}[S_1]| + |\mathbb{E}[D_1]| + |\mathbb{E}[T_1]| \right) + \epsilon \frac{1 - p^2}{p^2} \left( |\mathbb{E}[D_2]| + |\mathbb{E}[T_2]| \right) - \frac{1}{1 - 2p} \epsilon \mathbb{E}[Y_3] \leq \epsilon (n_0 + n_1 + 2n_2 + 7n_3) \leq 7 \epsilon (n_0 + n_1 + n_2 + n_3)$$

Similarly for other $X_i$’s, we get

$$\delta X_1 \leq 12 \epsilon (n_1 + n_2 + n_3)$$

$$\delta X_2 \leq 6 \epsilon (n_2 + n_3)$$

$$\delta X_3 \leq \epsilon n_3$$

Therefore, sampling every edge independently with probability $p$ satisfying all conditions in 40, all $X_i$’s concentrate within an additive gap of $(1 \pm 2\epsilon) n \left( \frac{1}{3} \right)$ with probability at least $1 - \frac{\epsilon}{m^\gamma}$. The constants in this proof can be tightened by a more accurate analysis.