Distributed Estimation of Graph 4-Profiles

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

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

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

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

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 every 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, distributed 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 [4, 21] as well as social and biological networks [29, 22]. 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, 29, 19].

There are 16 possible graphs on 3 vertices, labeled $H_0, \ldots, H_5$ in Figure 1a. 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 $\binom{n}{k}$, 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 [29, 31, 13, 7] and references therein).

In this paper we are interested in the significantly more challenging problem of estimating 4-profiles. Figure 1b shows the 11 possible graphs on 4 vertices, labeled $F_0, \ldots, F_{10}$. 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, with each coordinate 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.

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 the local 4-profiles may be rescaled and added together to obtain the global 4-profile.

Since some of the 4-profile subgraphs are disconnected (like $F_0, F_1, F_3$), 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 Actually there are 17 local subgraphs when considering vertex automorphisms. This is discussed in Section 2 in detail. For the purpose of initial exposition, we will ignore vertex automorphisms.
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. This restrictive setting does not allow communication between nonadjacent vertices, a key component of previous centralized, shared-memory approaches. Our algorithm relies on two novel theoretical results:

**Two-hop histograms are sufficient:** Our algorithm operates by having each vertex first perform local message-passing to its neighbors and then solve a novel system of equations for the local 4-profile. Focusing on a vertex $v_0$, the first easy step is to calculate its local 3-profile. It can be shown that the local 3-profile combined with the full two-hop connectivity information is sufficient to estimate the local 4-profile for each vertex $v_0$. This is not immediately obvious, since naively counting the 3-path (an automorphism of $F_4$) would require 3-hop connectivity information.

However, we show that less information needs to be communicated. Specifically, we prove that the triangle list combined with what we call the *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 $v_0$ that are 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) even more significant running time speedup of 5 to 10 times on several distributed experiments using 12 to 20 compute nodes.

**Profile Sparsification:** One idea that originated from triangle counting [27, 28] 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 rescale 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. Recently this idea was generalized to 3-profile sparsifiers in [7], with concentration results for estimating the full 3-profile. These papers rely on Kim-Vu polynomial concentration techniques [16] 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 global k-profile sparsifiers that use a novel information theoretic technique called read-k functions [11]. 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 PowerGraph [12] and tested it 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 5 million vertices and 40 million edges we estimated all local 4-profiles on this graph, the previous state of the art [13] required 1200 seconds while our distributed algorithm required less than 100 seconds.

1.2 Related Work

The problem of counting triangles in a graph has been addressed in distributed [24] and streaming [4] settings, and this is a standard analytics task for graph engines [23]. The Doulion algorithm [27] estimates a graph’s triangle count via simple edge subsampling. Other recent work analyzes more complex sampling schemes [25, 6] and extends to approximately counting certain 4-subgraphs [1, 15]. Mapreduce algorithms for clique counting were introduced by Finocchi et al. [10]. Our approach is similar to that of [7], which calculates all 3-subgraphs and a subset of 4-subgraphs distributedly using edge pivots. In this work we introduce the 2-hop histogram to compute all 4-subgraphs.

Concentration inequalities for the number of triangles in a random graph have been studied extensively. The standard method of martingale bounded differences (McDiarmid’s inequality) is known to yield weak concentrations around the mean for this problem. The breakthrough work of Kim and Vu [16] provides superior asymptotic bounds by analyzing the concentration of multivariate polynomials. This
was later improved and generalized in [14], and applied to
subsampled triangle counting in [28]. Our analysis uses a dif-
ferent technique called read-K functions [11] that produces
sharper concentration results for practical problem sizes.2

Previous systems of equations relating clique counts to
other 4-subgraphs appear in [17], [31], [13], and [2]. How-
ever, these are applied in a centralized setting and depend
on information collected from nonadjacent vertices. In this
work, we use additional equations to solve the same sys-
tem by sharing only local information over adjacent vertices.
The connected 4-subgraphs, or graphlets [22], have found
applications in fields such as bioinformatics [26] and com-
putational neuroscience [9]. In [30], authors use all global
4-subgraphs to analyze neuronal networks. We evaluate our
algorithm against Orca [13], a centralized 4-graphlet count-
ing algorithm, as well as its GPU implementation [20]. No-
tice that while Orca calculates only connected 4-subgraphs,
our algorithm calculates all the connected and the discon-
ected 4-subgraphs for each vertex.

Concurrent with the writing of this paper, a parallel algo-

algorithm for 4-subgraph counting was introduced in [2]. Our
algorithm differs by working within GraphLab PowerGraph’s
Gather-Apply-Scatter framework instead of the native, mul-
tithread C++ implementation of [2]. In terms of empirical
performance, both our work and [2] show similar running
time improvements of one order of magnitude over Orca.
A more detailed comparison would depend on the hardware
and datasets used. More importantly, our work focuses on a
distributed (as opposed to multicore parallel) framework,
and for our setting minimizing communication is critical.

Our theoretical results are significantly different from [2]
and may be useful in improving that system also. Specif-
ically, [2] explicitly counts the number of 4-cycles (F7 in
Figure 1b) whereas our results show that it is possible to
use only two-hop histograms instead. This results in less
communication overhead, but this benefit is perhaps not as
significant for shared-memory multicore platforms. Our sec-
ond theoretical result, the novel sparsification concentration
bounds, can be used for any subgraph estimation algorithm
and quantify a provable tradeoff between speed and accu-

2. 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 Power-
Graph framework [12] to get implicit connectivity informa-
tion about vertices outside the 1-hop neighborhood. Specif-
ically, 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 1. There is a distributed algorithm that com-
putes the exact local 4-profile of a graph given each vertex has
stored its local 3-profile, triangle list, and 2-hop histogram.

Note that the local 4-profiles at each vertex can be added
and appropriately rescaled (using the symmetries of each
subgraph, also called automorphism orbits [22]) to obtain
the global 4-profile.

4-Prof-Dist is implemented in the Gather-Apply-Scatter
(GAS) framework [12]. A distributed algorithm in this frame-
work 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 ⊕ (e.g., addition,
catenation). 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 the
algorithm’s completion.

4-Prof-Dist solves a slightly larger problem of keeping
track of counts of 17 unique subgraphs up to vertex auto-

4-Prof-Dist solves a slightly larger problem of keeping
track of counts of 17 unique subgraphs up to vertex auto-
morphism (see Figure 3). We will describe a full rank system
of equations which is sufficient to calculate the local 4-profile
at every v ∈ 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.

2.1 Edge Pivot Equations

The majority of our equations relate the local 4-profile
to neighboring local 3-profiles with edge pivots [7]. At each
vertex v, each combinatorial equation relates a linear com-

bination of the local 4-subgraph counts to the count of a
pair of 3-subgraphs sharing an edge va. Some of these equa-
tions appear in a centralized setting in previous literature
([17], [31], [13], [2]). In our algorithm, the 3-subgraph pair
count accumulates at v as all incident edges va pivot over it.
The edges fixed by a specific 3-subgraph pair correspond to
common edges among a subset of 4-subgraphs. Before that,
in an initial GAS round, each vertex v must gather the ID
of each vertex in its neighborhood, i.e. a ∈ Γ(v), and the

Figure 3: Unique (a) 3-subgraphs and (b) 4-
subgraphs from the perspective of the white vertex
v. F4 is the only subgraph with a third vertex au-
tomorphism F′, because no other subgraph contains
vertices with 3 different degrees.
following quantities must be stored at each edge \( ea \) during the Scatter phase:

\[
\begin{align*}
\text{n}_1^{e, va} &= |V| - (|\Gamma(v)| + |\Gamma(a)| - |\Gamma(v) \cap \Gamma(a)|), \\
\text{n}_2^{e, va} &= |\Gamma(v) \setminus (\Gamma(a) \cup a)| = |\Gamma(v)| - |\Gamma(v) \cap \Gamma(a)| - 1, \\
\text{n}_3^{e, va} &= \text{n}_2^{e, va}, \\
\text{n}_4^{e, va} &= |\Gamma(v) \cap \Gamma(a)|.
\end{align*}
\]

**Gather:** The above quantities are summed at each vertex \( v \) to calculate the local 3-profile at \( v \). For example, \( \text{n}_{3,v} = \frac{1}{2} \sum_{a \in \Gamma(v)} \text{n}_{3,va} \). In addition, we gather the sum of functions of pairs of these quantities forming 13 edge pivot equations:

\[
\begin{align*}
\sum_{a \in \Gamma(v)} \left( \frac{\text{n}_1^{e, va}}{2} \right) &= F_1(v) + F_2(v), \\
\sum_{a \in \Gamma(v)} \left( \frac{\text{n}_2^{e, va}}{2} \right) &= 3F'_8(v) + F'_8(v), \\
\sum_{a \in \Gamma(v)} \left( \frac{\text{n}_3^{e, va}}{2} \right) &= F'_8(v) + 3F_{10}(v), \\
\sum_{a \in \Gamma(v)} \text{n}_4^{e, va} v_a &= 2F'_3(v) + F'_6(v), \\
\sum_{a \in \Gamma(v)} \text{n}_5^{e, va} v_a &= 2F'_3(v) + F'_6(v), \\
\sum_{a \in \Gamma(v)} \text{n}_6^{e, va} v_a &= 2F'_3(v) + F'_6(v), \\
\sum_{a \in \Gamma(v)} \text{n}_7^{e, va} v_a &= 2F'_3(v) + F'_6(v), \\
\sum_{a \in \Gamma(v)} \text{n}_8^{e, va} v_a &= 2F'_3(v) + F'_6(v), \\
\sum_{a \in \Gamma(v)} \text{n}_9^{e, va} v_a &= 2F'_3(v) + F'_6(v), \\
\sum_{a \in \Gamma(v)} \text{n}_10^{e, va} v_a &= 2F'_3(v) + F'_6(v). \\
\end{align*}
\]

The primed notation differentiates between subgraphs of different automorphism orbits, as in (2). By accumulating pairs of 3-profile structures as in (2), we receive aggregate connectivity information about vertices more than 1 hop away. Consider the sixth equation as an example. The product between \( \text{n}_2^{e, va} \) and \( \text{n}_3^{e, va} \) subgraphs along edge \( va \) forms 4-node graphs with the following structural constraints: three vertex pairs are connected, two vertex pairs are disjoint, and one pair may be either connected or disjoint. \( F'_3(v) \) and \( F'_6(v) \) satisfy these constraints and differ on the unconstrained edge. Thus, as shown in Figure 4, they both contribute to the sum of \( \text{n}_2^{e, va} \) subgraphs.

The following edge pivot equations are linearly independent when solving for the local 4-profile only. Note the last 2 equations require calculating the local 3-profile:

\[
\begin{align*}
\sum_{a \in \Gamma(v)} \left( \frac{\text{n}_5^{e, va}}{2} \right) &= F_6(v) + F_8(v), \\
\sum_{a \in \Gamma(v)} \text{n}_1^{e, va} \text{n}_2^{e, va} &= F_3(v) + F_4(v), \\
\sum_{a \in \Gamma(v)} \text{n}_2^{e, va} \text{n}_3^{e, va} &= F'_8(v) + 2F_9(v), \\
\sum_{a \in \Gamma(v)} \text{n}_3^{e, va} - \text{n}_3^{e, va} &= F_8(v) + 2F_9(v) + 3F_{10}(v), \\
\sum_{a \in \Gamma(v)} \text{n}_5^{e, va} - \text{n}_5^{e, va} &= F_4(v) + 2F_7(v) + F'_8(v) + 2F'_9(v). \\
\end{align*}
\]

**Apply:** Store the left hand sides of all 13 equations at \( v \).
Algorithm 1 4-Pro\-f-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, $\cup_{e \in \Gamma(v)} a = \Gamma(v)$.
3: Apply: Store the gather as vertex data $v$.nb, size automatically stored.
4: Scatter: For each edge $e_{ua}$, compute and store scalars in (1).
5: Gather: For each edge $e_{ua}$, sum edge scalar data of neighbors in (2)- (3) and combine two-hop 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 $[v\cdot v.N0\cdot v.N1\cdot v.N2 \ldots \cdot v.N10]$}

transmitted across the network to a vertex $v$ is each non-neighboring vertex and its final count $\oplus_{u \in \Gamma(v)} c(p)$. Let

$$h_v = |\Gamma(\Gamma(v)) \setminus (\Gamma(v) \cup v)|.$$ 

For each $v$, the difference between full and histogram information is at most $\sum_{u \in \Gamma(v)} (|\Gamma(u)|) - 1 - 2h_v$. 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 ([3], 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.

2.4 Normalization and Symmetry

Our final local equation comes from summing the local 4-profile across all 17 automorphisms:

$$\sum_{i} F_i(v) = \binom{|V| - 1}{3}.$$ 

To calculate the global 4-profile, we utilize global symmetry and scaling equations. 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:

$$F_3 = 2F_1, \quad F_4 = F_2, \quad F_6 = 3F_6, \quad F_8 = F_8., \quad F'_8 = 2F_8, \quad F'_9 = F_9.$$ 

Global symmetry makes the equation for $F_8$ and the system (3) linearly dependent. We sum across vertices, inverting a single $11 \times 11$ system to yield the final global 4-profile $[N_0, \ldots, \ldots, N_{10}]$ by scaling appropriately:

$$N_0 = F_0, \quad N_1 = F_1, \quad N_2 = F_2, \quad N_3 = F_3,$$

$$N_4 = F_4, \quad N_5 = F_5, \quad N_6 = F_6, \quad N_7 = F_7,$$

$$N_8 = F_8, \quad N_9 = F_9, \quad N_{10} = F_{10}.$$ 

3. SPARSIFIER AND CONCENTRATION

In this section, we describe the process for approximating the exact number of subgraphs in a graph $G$. Denote the exact counts by $[N_0, \ldots, N_{10}]^T$ and the estimates by $[\tilde{N}_0, \ldots, \tilde{N}_{10}]^T$.

We are sparsifying the original graph $G$ by keeping each edge independently with probability $p$. Denote the random subsampled graph by $\tilde{G}$ and its global 4-profile by $[\tilde{N}_0, \ldots, \tilde{N}_{10}]^T$. Clearly each triangle survives with probability $p^3$ and each 4-clique survives with probability $p^4$. Therefore, in expectation, $E[\tilde{N}_{10}] = p^4 N_{10}$ and $\tilde{N}_{10}$ is unbiased.

This simple correspondence does not hold for other subgraphs: each triangle in $G$ can only be a triangle in $\tilde{G}$ that survived edge removals, but other subgraphs of $G$ 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-subgraph on the vertices $abcd$ before and after subsampling, respectively. Then define $H$ by $H_{ij} = P(F'(abcd) = F_i | F'(abcd) = F_j)$. Thus, we form an unbiased estimator, i.e. $E[\tilde{X}_i] = N_i, i = 1, \ldots, 10$, by inverting the edge sampling matrix.

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

$$\begin{bmatrix}
E[Y_0] \\
E[Y_1] \\
E[Y_2] \\
E[Y_3]
\end{bmatrix} =
\begin{bmatrix}
1 - p & (1 - p)^2 & (1 - p)^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
N_0 \\
N_1 \\
N_2 \\
N_3
\end{bmatrix}.$$ 

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.

3.1 Graph k-profile Concentration

Previous work used this idea of graph sparsification for triangle counting [28] and 3-profiles [7]. The main concentration tool used was the Kim and Vu polynomial concentration [16, 28] which unfortunately gives very loose bounds for practical graph sizes. Figure 5 compares the accuracy bound derived in this section to the bound predicted by [16]. Clearly the Kim-Vu concentration does not provide meaningful bounds for the experiments in Section 4.1. However, our results match observed sparsifier accuracy much more closely.

Our novel concentration results 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 [11]. 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. Additional details can be found in the extended version of this paper [8]. Our main concentration result is as follows:

**Theorem 2.** 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 $0 < \epsilon < 1$, and choose $R_K > 0$. If

$$p \geq \left( \frac{\log(2/\delta)k_{10}}{2 \sqrt[12]{2RKN_{10}}} \right)^{1/12},$$

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

**Proof.** Our proof relies on read-$k$ function families [11], a recent characterization of dependencies among functions of random variables. Rather than Lipschitz bounding the value of each partial derivative, as in [16, 28, 7], this technical tool benefits from the fact that each first partial derivative is sparse in the number of edge variables.

**Definition 1 (read-$k$ families).** Let $X_i, \ldots, X_m$ be independent random variables. For $j \in [r]$, let $P_j \subseteq [m]$ and let $f_j$ be a Boolean function of $X_j$. Assume that $\{|i| \in P_j\} \leq k$ 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. Let $G$ be a graph with $N_{10}$ 4-cliques and a maximum of $k_{10}$ 4-cliques sharing a common edge. The corresponding 4-clique estimator $X_{10}$ follows this exact structure. Each edge sampling variable appears in at most $k_{10}$ of the $N_{10}$ terms. We now state the main result required for our analysis. Note that when applied to estimating the number of 4-cliques, the bound is independent of the number of edges. Therefore, it is much stronger than arguments involving Lipschitz bounded functions such as McDiarmid’s inequality.

**Proposition 1 (Read-$k$ Concentration [11]).** Let $Z_1, \ldots, Z_r$ be a family of read-$k$ indicator variables with $P(Z_i = 1) = p_i$, and let $p$ be the average of $p_1, \ldots, p_r$. Then for any $\gamma > 0$,

$$P \left( \sum_{i=1}^r Z_i \geq (p + \gamma)r \right) \leq \exp \left( -D(p + \|p\|_K) \right),$$

$$P \left( \sum_{i=1}^r Z_i \leq (p - \gamma)r \right) \leq \exp \left( -D(p - \|p\|_K) \right),$$

where $D(x \| y) = x \log \left( \frac{x}{y} \right) + (1 - x) \log \left( \frac{1-x}{1-y} \right)$ is the Kullback-Leibler divergence of $x$ and $y$. Both bounds are less than $\exp(-2^{1/4}r/k)$.

Let $Y_{10} = \sum_{(a,b,c,d) \in H_{10}} t_{ab}t_{bc}t_{cd}t_{da}t_{ac}t_{db}$. Then

$$P \left( |Y_{10} - p^5N_{10}| \geq \sqrt{RKN_{10}} \right) \leq 2 \exp \left( -\frac{2^{1/12} \sqrt{2RKN_{10}}}{k_{10}} \right),$$

$$\Rightarrow P \left( |X_{10} - N_{10}| \geq \sqrt{RKN_{10}} \right) \leq 2 \exp \left( -\frac{2^{12} \sqrt{2RKN_{10}}}{k_{10}} \right).$$

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

Next, we state conditions under which our method outperforms the Kim and Vu concentration results [16].

**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.

**Proof.** 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 [16, 28, 7].

**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 $a_0 = 8^6\sqrt{6}l$, $0 < p \leq 1$, and $l_K > 0$. Let $X_{10}$ be the 4-clique estimate obtained from subsampling each edge with probability $p$. If

$$\frac{p}{\max \left\{ \sqrt{k_{10}N_{10}} \right\}} \geq \frac{a_0^2 \log^{12}(m^{2+\gamma})}{k_K},$$

then $|N_{10} - X_{10}| \leq \sqrt{kVN_{10}}$ with probability at least $1 - \frac{1}{m^7}$.

The proof of this lemma is a straightforward application of the main result in [16]. It can be found in the extended version of this paper [8].

Now we are ready to prove the corollary by comparing Theorem 2 and Lemma 1. Fix $p, \delta, \gamma > 0$. For $\gamma > 1$ such that $p = \Omega(1/\log m)$ and $\delta = \Omega(1/m)$. Now we analyze the bounds $l_K$ and $R_K$. For any graph and $a_0$ defined in Lemma 1,

$$\frac{l_K}{a_0^2} \leq 1, \quad \frac{\gamma}{(5 + \gamma)^{12}} \leq 1, \quad \frac{\log(2^{1/4}m)}{2 \log m} \leq 1.$$ (10)

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

$$p^{11} \geq \frac{1}{\log^{11}(m)} \geq \frac{\log^{12}(m)}{2a_0^2} \min \left\{ \frac{k_{10}N_{10}^{5/6}}{N_{10}}, \frac{(k_{10}N_{10})^{3/2}}{128} \right\}.$$

Rearranging terms,

$$\frac{\gamma}{2k_K} = \frac{\log(2m^{2+\gamma})}{2p^{12}} \max \left\{ \frac{\sqrt{k_{10}N_{10}}}{k_K} \right\}.$$}

We note that the asymptotic condition on $p$ in Corollary 1 includes a constant term much less than 1. This is due to the looseness of inequalities in (10) and implies that Theorem 2 is superior to Lemma 1 over all $p$ values of practical interest. While these bounds contain the quantities we wish to estimate, they provide guidelines for the performance of sampling heuristics. We also investigate this in Section 4.1 for some realistic graphs.
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. 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 cannot 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.](image)

### 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 ±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.

**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.

---

3Available at http://github.com/eelenberg/4-profiles
4Amazon Web Services - http://aws.amazon.com
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 8: Running time of 4-Prof-Dist for various number of compute nodes and sampling probability $p$, on the LiveJournal graph.

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

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 $H$ is defined by the relations

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

$$H^{-1} = \left[ \begin{array}{c}
S_{11} & S_{12} \\
0_{p \times 7} & S_{22}
\end{array} \right], \text{ where}

S_{11} = \\
\left[ \begin{array}{cccccc}
t & \frac{t^2}{7} & \frac{t^3}{7} & \frac{t^4}{7} & \frac{t^5}{7} & \frac{t^6}{7} \\
0 & \frac{t}{7} & \frac{t^2}{7} & \frac{t^3}{7} & \frac{t^4}{7} & \frac{t^5}{7} \\
0 & 0 & \frac{t}{7} & \frac{t^2}{7} & \frac{t^3}{7} & \frac{t^4}{7} \\
0 & 0 & 0 & \frac{t}{7} & \frac{t^2}{7} & \frac{t^3}{7} \\
0 & 0 & 0 & 0 & \frac{t}{7} & \frac{t^2}{7} \\
0 & 0 & 0 & 0 & 0 & \frac{t}{7}
\end{array} \right]$$

S_{12} = \\
\left[ \begin{array}{cccccc}
t^2 & \frac{t^3}{7} & \frac{t^4}{7} & \frac{t^5}{7} & \frac{t^6}{7} \\
\frac{t^2}{7} & \frac{t^3}{7} & \frac{t^4}{7} & \frac{t^5}{7} & \frac{t^6}{7} \\
\frac{t^2}{7} & \frac{t^3}{7} & \frac{t^4}{7} & \frac{t^5}{7} & \frac{t^6}{7} \\
\frac{t^2}{7} & \frac{t^3}{7} & \frac{t^4}{7} & \frac{t^5}{7} & \frac{t^6}{7} \\
\frac{t^2}{7} & \frac{t^3}{7} & \frac{t^4}{7} & \frac{t^5}{7} & \frac{t^6}{7} \\
\frac{t^2}{7} & \frac{t^3}{7} & \frac{t^4}{7} & \frac{t^5}{7} & \frac{t^6}{7}
\end{array} \right]$$

S_{22} = \\
\left[ \begin{array}{cccccc}
\frac{1}{p^2} & 0 & \frac{1}{p^2} & \frac{1}{p^2} & \frac{1}{p^2} & \frac{1}{p^2} \\
0 & \frac{1}{p^2} & \frac{1}{p^2} & \frac{1}{p^2} & \frac{1}{p^2} & \frac{1}{p^2} \\
0 & 0 & \frac{1}{p^2} & \frac{1}{p^2} & \frac{1}{p^2} & \frac{1}{p^2} \\
0 & 0 & 0 & \frac{1}{p^2} & \frac{1}{p^2} & \frac{1}{p^2}
\end{array} \right],$$

and $0_{4 \times 7}$ is a $4 \times 7$ 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 $X_i$, $0 \leq i \leq 9$ (and $X$ 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 $Y_i$, $0 \leq i \leq 10$, be the 4 profile counts of the sparsified graph. Then let $N_i$, $0 \leq i \leq 10$, be the actual counts. Choose $0 < \delta < 1$ and $\epsilon > 0$. Let $C = (192)^2/2$ and

$$k_a = k_2 + k_3, \quad k_b = k_4 + k_5 + k_6, \quad k_y = k_7 + k_8, \quad N_a = N_2 + N_3, \quad N_b = N_4 + N_5 + N_6, \quad N_y = N_7 + N_8.$$
The $Y_i$'s have the following parameters:

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

We apply Proposition 1 to each estimator. This is shown in the proof of Theorem 2 for $Y_0$ and in the extended paper [8] for the other estimators. Rearranging to solve for $p$,

\[
p \geq \left(\frac{\log(2/\delta)}{2^{-2}(N_{10})}\right)^{1/10} + \left(\frac{\log(2/\delta)}{2^{-2}(N_{10})}\right)^{1/8},
\]

where

\[
k_0 = k_2 + k_4, \quad k_0 = k_4 + k_5 + k_6, \quad k_0 = k_7 + k_8, \quad N_a = N_2 + N_3, \quad N_{10} = N_4 + N_5 + N_6, \quad N_{10} = N_7 + N_8.
\]

The final condition comes from the result for $Y_0$:

\[
n_0 \leq \left(\frac{|V|}{4}\right) - \frac{\log(2/\delta)}{2^{-2}(N_{10})} \leq |V| - \frac{\log(2/\delta)}{2^{-2}(N_{10})}.
\]

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

\[
\begin{align*}
\delta X_0 &\leq (n_1 + n_2 + n_3) + (n_1 + 2n_2 + 3n_3 + n_2 + 3n_4 + n_3) \\
&\leq (2n_1 + 4n_2 + 8n_3) \leq 8 \left(\frac{|V|}{3}\right) \\
\delta X_1 &\leq (N_1 + \ldots + 192N_{10}) \leq 192 \left(\frac{|V|}{4}\right) \\
\delta X_2 &\leq (N_2 + \ldots + 48N_{10}) \leq 48 \left(\frac{|V|}{4}\right) \\
\delta X_3 &\leq (N_3 + 4N_4 + 6N_5 + \ldots + 192N_{10}) \leq 192 \left(\frac{|V|}{4}\right) \\
\delta X_4 &\leq (N_4 + \ldots + 96N_{10}) \leq 96 \left(\frac{|V|}{4}\right) \\
\delta X_5 &\leq (N_5 + \ldots + 32N_{10}) \leq 32 \left(\frac{|V|}{4}\right) \\
\delta X_6 &\leq (N_6 + \ldots + 32N_{10}) \leq 32 \left(\frac{|V|}{4}\right) \\
\delta X_7 &\leq (N_7 + 2N_8 + 12N_{10}) \leq 12 \left(\frac{|V|}{4}\right) \\
\delta X_8 &\leq (N_8 + 4N_9 + 12N_{10}) \leq 4 \left(\frac{|V|}{4}\right) \\
\delta X_9 &\leq (N_9 + 6N_{10}) + 6 \left(\frac{|V|}{4}\right) \\
\delta X_{10} &\leq N_{10}.
\end{align*}
\]

Thus the maximum deviation in any estimator is less than 192 $\left(\frac{|V|}{4}\right)$. Substituting $\gamma = 2/192$ completes the proof. \[\square\]