Counting Graphlets: Space vs Time

Marco Bressan*
Dipartimento di Informatica
Sapienza University of Rome
bressan@di.uniroma1.it

Flavio Chierichetti†
Dipartimento di Informatica
Sapienza University of Rome
flavio@di.uniroma1.it

Ravi Kumar
Google Research
Mountain View, CA
ravi.k53@gmail.com

Stefano Leucci*
Dipartimento di Informatica
Sapienza University of Rome
leucci@di.uniroma1.it

Alessandro Panconesi*
Dipartimento di Informatica
Sapienza University of Rome
ale@di.uniroma1.it

ABSTRACT

Counting graphlets is a well-studied problem in graph mining and social network analysis. Recently, several papers explored very simple and natural approaches based on Monte Carlo sampling of Markov Chains (MC), and reported encouraging results. We show, perhaps surprisingly, that this approach is outperformed by a carefully engineered version of color coding (CC) [1], a sophisticated algorithmic technique that we extend to the case of graphlet sampling and for which we prove strong statistical guarantees. Our computational experiments on graphs with millions of nodes show CC to be more accurate than MC. Furthermore, we formally show that the mixing time of the MC approach is too high in general, even when the input graph has high conductance. All this comes at a price however. While MC is very efficient in terms of space, CC's memory requirements become demanding when the size of the input graph and that of the graphlets grow. And yet, our experiments show that a careful implementation of CC can push the limits of the state of the art, both in terms of the size of the input graph and of that of the graphlets.

1. INTRODUCTION

Counting graphlets is a well-studied problem in graph mining and social-networks analysis [2, 5, 6, 9, 12, 16, 22–24, 26]. Given an input graph, the problem asks to count the frequencies of all induced connected subgraphs (called *graphlets*), up to isomorphism, of a certain size. This problem is highly motivated in the context of studying behavioral and biological networks. Understanding the distribution of graphlets allows us to make key inferences about the structural properties of the underlying graph and the interaction of the nodes in the graph (e.g. [18]). It sheds light on the type of local structures that are present in the graph, which can be used for a myriad of analysis [2, 6, 13, 22–24]. For example, an extreme case of graphlets is the three-node graphlet: counting triangles is a fundamental problem that has been repeatedly studied for the insights it can yield about the health of a network and also for pushing the boundaries of computation that is possible with large networks [11, 17, 19]. How the graphlets form in the first place and how they temporally evolve are semantically more actionable than the interpretation yielded by the mere evolution of nodes and edges.

Since the exact counting of graphlets can be computationally demanding, one usually settles for less ambitious goals. One such goal, and the one we pursue in this paper, is frequency estimation: for each subgraph we want to estimate, as accurately as possible, its relative frequency among all subgraphs of the same size. Less ambitiously still, since the number of subgraphs of a given size grows exponentially, we will be interested in the problem of estimating the relative frequency of only the most frequent ones, say, those that appear at least a certain fraction of the time.

There have been two popular approaches to obtaining such estimates. The first is to use Markov Chain Monte Carlo (henceforth, MC). Given an input graph, consider a Markov chain whose nodes (states) are the graphlets, and where two graphlets are connected if they differ by a node. After adding an appropriate number of self-loops to make the chain regular, it follows from standard facts that a random walk of length equal to (or greater than) the mixing time will stop at a uniform node (i.e., a graphlet). This gives a very simple and space-efficient way to sample the population of graphlets. Just repeat the walk independently very many times. Recently this approach has been tried by several authors with encouraging results [2, 9, 16, 22]. For this type of approach to be statistically reliable however, the crucial question is, how long is the mixing time of this natural walk? To the best of our knowledge, this question has not been addressed in a principled manner.

A second approach to count graphlets (especially, trees) is to use color coding (CC), an elegant randomized algorithmic technique introduced in [1]. CC provides strong, provable statistical guarantees for the problem of approximating exact graphlet counts, from which the frequencies can be easily derived. From a computational point of view, perhaps its main drawback is its space requirement, that can quickly become insurmountable as the graphlet size grows. Furthermore, for its nice statistical guarantees to hold in the case of graphlets, one needs to run CC an exponential (in the subgraph size) number of times, which can be prohibitive. This heavy price must be paid if one needs precise estimates of exact counts. But what if one is just interested, as we are in this paper, in estimating the frequency of the most common graphlets, can a linear upper bound be attained?

*Supported in part by a Google Focused Research Award, by the Sapienza Grant C26M15ALKP, by the SIR Grant RBSI14Q743, and by the ERC Starting Grant DMAP 680153.

†Work done in part while visiting Google. Supported in part by a Google Focused Research Award, by the ERC Starting Grant DMAP 680153, and by the SIR Grant RBSI14Q743.

WSDM 2017, February 06-10, 2017, Cambridge, United Kingdom

DOI: http://dx.doi.org/10.1145/3018661.3018732

Our contributions. In this work, we study MC and CC as the most viable methods for counting reasonably sized graphlets in massive graphs. Our goal is to understand and compare these methods from a practical point of view and en route show provable guarantees. Let n be the size of the input graph and let k be the size of graphlet.

Our first contribution is to study the mixing time of MC. We show that even if the input graph is well mixing (as most social networks are) and even if there is one subgraph that appears more that 99% of the time, it is possible that MC will take $\Omega(n^{k-1})$ steps before sampling the most frequent graphlet just a single time! Note that this is not far from the naive $O(n^k)$ bound needed to perform exact counting by an exhaustive enumeration. In particular, this shows that the mixing time of MC can be huge even when that of the input graph is very small, a somewhat counterintuitive statement. For large graphs and even modest values of k, this readily implies that, unless one makes specific assumptions about the input graph and exploits them in the analysis, the MC approach, which has been used in several past work, effectively gives no statistical guarantees. On the positive side, we show that the mixing time of MC is $O(n^2\Delta^{2k})$, a bound that can be useful for graphs of moderate size and small maximum degree Δ.

Our next contribution is to study the effectiveness of CC for graphlet counting. In CC, which is basically a dynamic program, there are two stages: a building phase and a sampling phase. The building phase is a time and space consuming process and is in a sense, inevitable. The sampling phase, however, can become a bottleneck for large k. To address this problem we propose a new version of CC that trades off the space in the building phase versus the time in the sampling phase; this enables us to count graphlets for large k. We then show that even a single run of CC, whose output can be seen as a large sample of the population of graphlets, gives reasonably good statistical guarantees. We remark that these bounds are still too weak to provide strong confidence in realistic situations. But, at the very least, they offer some evidence that by compounding the estimates obtained with very few runs of CC one can get good perhaps even strong statistical guarantees. We view our result as an encouraging step along this direction, a line of research that we believe is an interesting one.

It is often the case that theoretical bounds are too coarse, while in actuality algorithms are much better behaved. Indeed, this seems to be the case with CC, which we put to test using real-world graphs with several millions of nodes and various values of k. We consistently observe that the estimate given by just a single bag of graphlets is comparable to that obtained by averaging many runs, an outcome that is in line with the considerations above.

CC and MC represent, to the best of our knowledge, the state-of-the-art in graphlet counting. We therefore carry out an experimental analysis with real-world graphs to assess their performance in a comparative framework when the goal is estimating the frequencies of the most common graphlets. Our experiments are performed with graphs whose sizes vary from small to several millions of nodes, and values of $k = 3, \ldots, 7$. In a nutshell, the outcome is that the two approaches are comparable in terms of running time. More crucially, CC is superior in terms of statistical guarantees, but is also much more demanding in terms of space. As a rule of thumb, for critical applications in our opinion CC remains preferable, while MC becomes competitive in the remaining cases.

2. PRELIMINARIES

A graph $G = (V, E)$ is composed of a set V of nodes and a set $E \subseteq \binom{V}{2}$ of edges[1]. In this paper we will assume the graph is connected and undirected.

The *degree* of a node $v \in V$ is the number of nodes w such that $\{v, w\}$ is an edge of G: $\deg_G(v) = \{w \mid \{v, w\} \in E\}$. We use $\Delta(G)$ to denote the maximum degree of G: $\Delta(G) = \max_{v \in V} \deg_G(v)$. When G is obvious from the context, we simply use $\deg(\cdot)$ and Δ.

Graphlets. Given a graph $G = (V, E)$, and a subset $W \subseteq V$ of its nodes, we let the subgraph of G induced by W, or $G|W$, be the graph composed of the set of nodes W and the set of edges $E \cap \binom{W}{2}$. If $|W| = k$ and if $G|W$ is connected, then we say that $G|W$ is a *k-graphlet* of G. We denote by $\mathcal{V}_k(G)$ the set of k-graphlets of G. Finally, if H is a connected undirected graph on k nodes, we say "the number of occurrences of H in G" to refer to the number of elements in $\mathcal{V}_k(G)$ that are isomorphic to H.

3. GRAPHLETS VIA COLOR CODING

In this section we present two algorithms for graphlet counting, which are based on color coding (CC), a powerful algorithmic technique introduced by Alon, Yuster, and Zwick [1]. Given the input graph $G = (V, E)$ and k, coloring coding first assigns uniformly and independently to each node of G a random label in $[k] := \{1, \ldots, k\}$, referred to as a *color*. The goal now is to count the number of *non-induced* trees of k nodes in G—called *treelets*—that are *colorful*, i.e., whose labels have no repetitions. This can be done efficiently by dynamic programming, thanks to the fact that treelets with disjoint set of labels must lie on disjoint set of nodes. Since a treelet is colorful with relatively low probability, one just needs to repeat the coloring sufficiently many times in order to "hit" any given treelet.

Color coding requires time $O(c^k \cdot |E|)$ and space $O(c^k \cdot |V|)$ for some $c > 1$, which has made it possible to push the task of estimating subgraph counts in the realm of graphs with millions of nodes. However, all known algorithms can only count subgraphs occurrences that are not induced, and that are either trees or "tree-like" in the sense of having low treewidth. We show that treelet counts can be extended to graphlet counts based on the observation that by counting treelets we have counted (with high probability if repeated many times) all the spanning trees of every graphlet. To summarize, a good estimate of treelets can be translated into a good estimate of graphlets.

In this section we first describe the two algorithms that are based on color coding. The first algorithm is the one of Alon et al. [1] and the second is a modification of this algorithm to use more space but faster in terms of running time. We then prove concentration on the number of colorful treelets produced by one run of (either of these) algorithms. We will use this to prove concentration on the number of colorful graphlets.

3.1 Algorithms

Here we describe two algorithms based on color coding that can count and sample colorful non-induced treelets uniformly at random. We then show how that suffices to sample colorful induced graphlets, as well. Both algorithms start with a coloring phase where each node $v \in V$ of G is assigned a color $c(v)$ chosen independently and uniform at random from $[k]$.

3.1.1 The first algorithm (CC1)

The first algorithm is that of Alon et al. [1], which we describe for completeness. In this algorithm (called CC1), each possible rooted tree T on $h \leq k$ nodes is considered and, for each node $v \in V$ and set $S \subseteq [k]$ of exactly h colors, the number $C(T, S, v)$

[1]For a finite set S and an integer $0 \leq k \leq |S|$, we use $\binom{S}{k}$ to denote the set of k-subsets of S, i.e., $\binom{S}{k} = \{T \mid T \subseteq S, |T| = k\}$.

of occurrences of (non-induced) treelets rooted at v that are isomorphic to T and whose node labels span the set S is counted. This is done as follows: if $|S| = 1$ then $C(T, S, v) = 1$ if $c(v)$ coincides with the unique color in S, and $C(T, S, v) = 0$ otherwise. If $|S| \geq 2$, then T is split into two rooted subtrees T_1 and T_2 by removing any e incident to the root of T (the endpoints of the edge become the root of the subtrees), and $C(T, S, v)$ is computed using the following relation:

$$C(T, S, v) = \frac{1}{d} \sum_{(v,u)\in E} \sum_{\substack{S_1, S_2 \subset S \\ S_1 \cap S_2 = \emptyset}} C(T_1, S_1, v) \cdot C(T_2, S_2, u),$$

where d is a normalization constant that is equal to the number of rooted trees isomorphic to T_2 among the subtrees rooted in the children of the root T.

Once all the values $C(T, S, v)$ have been computed, it is possible to sample a treelet on k nodes of G uniformly at random. In order to do so, we first randomly choose a node v of G with probability proportional to the overall number of occurrences of treelets of size k rooted at v (i.e., the sum of the quantities $C(T', \cdot, v)$ where T' has k nodes). Then, we pick a treelet T and a set S proportionally to the values $C(T, S, v)$. We now need to choose one of the $C(T, S, v)$ treelets rooted at v that are isomorphic to T and are colored with the colors in S. To this aim we split T into T_1 and T_2 as described above and we simultaneously choose (i) a neighbor u of v and (ii) the two sets S_1 and S_2 of colors such that $S_1 \cap S_2 = \emptyset, S_1 \cap S_2 = S$, proportionally to the quantity $C(T_1, S_1, v) \cdot C(T_2, S_2, u)$. Then we recursively sample one of the $C(T_1, S_1, v)$ (resp. $C(T_2, S_2, v)$) treelets isomorphic to T_1 (resp. T_2) and colored with the colors in S_1 (resp. S_2) from v (resp. u). The following is easy to show (proof omitted).

THEOREM 1. *Algorithm CC1 uses time $O(c^k|E|)$ and space $O(c^k|V|)$ and generates a treelet sample in time $O(c^k)$ for some constant $c > 1$.*

3.1.2 The second algorithm (CC2)

In this modification, we consider *colored* rooted trees T. The algorithm (called CC2) counts the number $C(T, v)$ of occurrences of non-induced treelets rooted at v such that the colors of the occurrence exactly match the colors of T. Each tree T rooted in r with more than 1 node can be split in an unique way into two trees T_1 and T_2 where T_2 is the (colored) subtree of T rooted in the child of r having the highest color, and T_1 is the remaining tree. Similarly, two trees T_1 and T_2 can be merged iff the color of the root of T_2 is larger than the largest color of a children of the root of T_1. Merging T_1 and T_2 results in the tree T. If T has $h = 1$ node, then $C(T, v) = 1$ if v matches its color and $C(T, v) = 0$ otherwise. Once all the values of $C(\cdot, \cdot)$ for trees up to size $h - 1$ have been computed, we can also compute all the values $C(T, v)$ where T with has h nodes as follows: set $C(T, v) = 0$ for all the trees T on h nodes, then for each edge (u, v) of the graph and for each pair of trees T_1 and T_2 such that $C(T_1, u) \neq 0$ and $C(T_1, v) \neq 0$, check if T_1 and T_2 can be merged into a tree T. If that is the case, add $C(T_1, u) \cdot C(T_2, v)$ to $C(T, u)$.[2]

To sample a treelet of k nodes we first pick a node v of G with probability proportional to the sum of values $C(\cdot, v)$. Then we pick a rooted tree T with a probability proportional to to $C(T, v)$. Finally, we split T into T_1 and T_2 as shown above, we choose a neighbor u of v in G with probability proportional to $C(T_1, v) \cdot C(T_2, v)$, and we recursively select one occurrence of T_1 from v and one occurrence of T_2 from u. The following is easy to show (proof omitted).

[2]Here an undirected edge between u and v is considered two times: both as (u, v) and as (v, u).

THEOREM 2. *Algorithm CC2 uses space $O(k^k|V|)$ and generates a treelet sample in time $O(\log |V|)$.*

3.1.3 From treelets to graphlets

Once we can sample an occurrence of a colorful treelet on k uniform at random from the set of all treelet occurrences in G, it is possible to also sample a colorful (induced) graphlet H by noticing that each spanning tree of H gets counted *exactly* k times by the previous algorithms. Hence, the algorithm to sample a graphlet looks like:

(i) sample an occurrence T of a treelet on k nodes from G

(ii) consider the graphlet H induced by the nodes of T, and

(iii) reject H with probability $1 - \frac{1}{k\sigma(H)}$, where $\sigma(H)$ is the number of spanning trees of H.

The value $\sigma(H)$ can be (pre)computed for all H in time $O(c^k)$, e.g., via the Kirchhoff's Matrix-Tree Theorem [20].

3.1.4 Practical considerations

Notice that CC1 can also be implemented by considering one edge of G at time for each value of h (somewhat similar to CC2) by consistently picking the edge e of T to be removed in the splitting step: two trees T_1 and T_2 rooted in v and u can be merged if (i) their color-sets are disjoint and (ii) the edge e of the tree T obtained by appending T_2 as a child of the root of T_1 coincides with (v, u).

This equivalent implementation allows us to consider, for each edge (v, u) only the pairs of treelets T_1, T_2 whose values $C(v, T_1)$ and $C(u, T_2)$ are not 0. We use this implementation for both CC1 and CC2.

Moreover, to speed up the sampling steps we precompute the distributions needed to choose the first node v, the tree T, and the set S for CC1 (resp. the first node v and the tree T for CC2) and use inverse transform sampling. We did not, however, precompute the distribution allowing us to (recursively) select the root of T_1. Since the average degree of the graphs is small, we can quickly construct those distributions on demand without a significant increase in the sampling time. This allows us to reduce the overall memory footprint of the algorithms.

3.2 Concentration of colored graphlets

In this section we show tight concentration bounds on the number of colorful graphlets produced by CC1 and CC2, or more precisely, by any random uniform coloring of the nodes of G. We henceforth assume G is connected and $k \geq 3$, and denote by $g = |\mathcal{V}_k(G)|$ the total number of k-graphlets in G. Putting ourselves in a more general scenario, suppose we are interested in estimating the size of an arbitrary subset of k-graphlets of G; it can be again the set $\mathcal{V}_k(G)$ of all k-graphlets, or the set of all k-graphlets isomorphic to some H, etc. Our main result is the following:

THEOREM 3. *Consider any $\mathcal{S} \subseteq \mathcal{V}_k(G)$, and let $Z_\mathcal{S}$ be the random variable counting the number of graphlets in $\mathcal{S}$ that are made colorful by a coloring of G. Let $s = |\mathcal{S}|$ and $\mu_\mathcal{S} = \mathbb{E}[Z_\mathcal{S}]$. Then for any $\epsilon > 0$:*

$$\Pr\left[\, |Z_\mathcal{S} - \mu_\mathcal{S}| > \epsilon\, \mu_\mathcal{S} \,\right] \leq e^{-\Omega(\epsilon^2 s^{1-1/k}/g^{1-2/k})},$$

where the $\Omega(\cdot)$ notation hides factors that depend only on k but not on g and s.

The proof of Theorem 3 is deferred to the next subsection, and can be skipped without impairing the understanding of the rest of the paper. Let us make some considerations on the above bound. First

of all, when $\mathcal{S} = \mathcal{V}_k(G)$, i.e., when we are observing the number of overall colorful k-graphlets in G, the bound simplifies to:

COROLLARY 4. *The number Z of k-graphlets of G that are made colorful by a random coloring satisfies, for any $\epsilon > 0$:*

$$\Pr\left[\, |Z - \mathbb{E}[Z]| > \epsilon\, \mathbb{E}[Z] \,\right] \leq e^{-\Omega(\epsilon^2 g^{1/k})}.$$

This form lends itself to an interesting interpretation: the exponent is driven by $g^{1/k}$, which is essentially the minimum number of distinct nodes needed to form the g graphlets in the graph (this happens if those nodes form a clique). This implies tightness of the bounds, as formalized next:

THEOREM 5. *For any real $\alpha \in [1, k]$ there exist arbitrarily large graphs on n nodes containing $g = \Theta(n^\alpha)$ graphlets such that for any $\epsilon > 0$, $\Pr[|Z - \mathbb{E}[Z]| > \epsilon\, \mathbb{E}[Z]] \geq e^{-O(g^{1/k})}$, where Z counts the total number of colorful graphlets.*

PROOF. Consider a graph formed by a clique on $\ell = \Theta(n^{\alpha/k})$ nodes linked to a path on $n-\ell$ nodes. The total number of graphlets is clearly $g = \Theta(\ell^k) = \Theta(n^\alpha)$. Now choose ℓ large enough so that the clique contains more than $\epsilon\, \mathbb{E}[Z]$ graphlets (this is always possible since $\alpha \geq 1$). With probability $k^{1-\ell} = e^{-O(g^{1/k})}$, all the clique nodes have the same color and $Z < (1-\epsilon)\mathbb{E}[Z]$. □

In general, the exponent of the bound of Theorem 3 can fall to $o(g^{1/k})$, but remains $\omega(1)$ as long as $s \in \Omega(g^{(k-2)/(k-1)})$, i.e., as long as $\mathcal{S}$ is large enough, which covers many cases of practical interest such as counting the most frequent graphlets. And also in this sense the bound is tight:

THEOREM 6. *There exist arbitrarily large graphs where the set $\mathcal{S}$ of graphlets not spanned by a star has size $\Omega(g^{(k-2)/(k-1)})$ and yet $\Pr[Z_\mathcal{S} = 0] \geq \frac{1}{k}$.*

PROOF. Consider a graph on n nodes formed by a star on $n-1$ nodes, plus one additional node u attached to a node u' that is a leaf of the star. There are $\binom{n-1}{k-1} = \Omega(n^{k-1})$ graphlets isomorphic to stars, thus $g = \Omega(n^{k-1})$. The set $\mathcal{S}$ of graphlets not spanned by stars contains all and only those graphlets containing both u and u', which are $\binom{n-2}{k-2} = \Omega(n^{k-2}) = \Omega(g^{(k-2)/(k-1)})$. The probability that *all* such graphlets are not colorful, and thus that $Z_\mathcal{S} = 0$, is no smaller than $\Pr[u \text{ and } u' \text{ have the same color}] \geq \frac{1}{k}$. □

By Theorem 3, the distribution of colorful graphlets (which can be sampled via our algorithms CC1 and CC2) closely matches the overall distribution of graphlets, at least for graphlets that occur often enough. A formal statement is the following (proof omitted):

COROLLARY 7. *Let $\mu_H \in [0, 1]$ be the fraction of graphlet occurrences of G that are isomorphic to H, and let S be a graphlet drawn uniformly at random from the set of colorful graphlets of G. Then for any $\epsilon > 0$, with probability $1 - e^{-\Omega(\epsilon^2 g^{1/k})}$:*

$$\Pr[S \text{ is an occurrence of } H] = \mu_H \pm \frac{\epsilon}{1-\epsilon}.$$

Finally, we claim (proof omitted) that, if one creates $\lambda \geq 1$ random colorings of G, and takes the average counts of each graphlet, one gets the following improved concentration bound.

COROLLARY 8. *If we consider λ independent colorings of G and let $Z_\mathcal{S} = \lambda^{-1} \sum_{i=1}^{\lambda} Z_\mathcal{S}^i$ where $Z_\mathcal{S}^i$ counts the number of colorful graphlets of $\mathcal{S}$ in the ith coloring, then the bound of Theorem 3 becomes:*

$$\Pr\left[\, |Z_\mathcal{S} - \mu_\mathcal{S}| > \epsilon \cdot \mu_\mathcal{S} \,\right] \leq e^{-\Omega(\lambda \epsilon^2 s^{1-1/k} / g^{1-2/k})}.$$

3.2.1 Proof of Theorem 3

The key steps of the proof are as follows. We assume the nodes of G are colored in nonincreasing order of number of graphlets they appear in (any order is clearly equivalent). We then consider the (Doob) martingale that counts the expected number of colorful graphlets in $\mathcal{S}$ given the colors assigned to the first i nodes, for each $i = 1, \ldots, n$. By applying the method of bounded differences, we get a concentration inequality whose exponent's denominator has one term for each node of G, telling how much the martingale can oscillate when we color that node. Thanks to the ordering of the nodes, bounding the vast majority of terms due to nodes that appear in few graphlets is relatively easy; less so for the other terms, that also depend on how many graphlets can be shared by two nodes. This requires us to prove that two nodes cannot simultaneously appear in too many graphlets (one of the two must appear in asymptotically more).

Let us start with some notation. For $i = 1, \ldots, n$, let $X_i \in [k]$ be the random variable denoting the color of node i. For $j = 1, \ldots, s$ let $Y_j \in \{0, 1\}$ be the indicator random variable of the event that the jth graphlet of $\mathcal{S}$ is colorful, and let $Z = Z_\mathcal{S} = \sum_{j=1}^{s} Y_j$ be the total number of colorful graphlets of $\mathcal{S}$. Finally, for $i = 1, \ldots, n$ let $Z_i = \mathbb{E}[Z|X_1, \ldots, X_i]$ be the expectation of Z as a function of the colors assigned to the first i nodes, and let $Z_0 = \mathbb{E}[Z]$. The sequence $Z_0, \ldots, Z_n$ is a Doob martingale with respect to $X_1, \ldots, X_n$, and Azuma's inequality implies

$$\Pr\left(\, |Z - \mathbb{E}[Z]| > t \right) < 2e^{-\frac{t^2}{2\sum_{i=1}^{n} c_i^2}}, \tag{1}$$

whenever $|Z_i - Z_{i-1}| \leq c_i$. Let now $g_\mathcal{S}(u)$ be the number of graphlets of $\mathcal{S}$ in which u appears. The rest of the proof is devoted to showing that, if the nodes of G are sorted in nonincreasing order of $g_\mathcal{S}(i)$, then $\sum_{i=1}^{n} c_i^2 = O(s^{1+\frac{1}{k}} / g^{1-\frac{2}{k}})$, which together with Equation 1 implies Theorem 3 for $t = \epsilon s$.

Start by breaking $\sum_{i=1}^{n} c_i^2$ in two parts:

$$\sum_{i=1}^{n} c_i^2 = \sum_{i=1}^{\ell} c_i^2 + \sum_{i=\ell+1}^{n} c_i^2, \tag{2}$$

for some ℓ to be chosen later. Note that $c_i \leq g_\mathcal{S}(i)$: conditioning on X_i can alter, by at most 1, the expectation of only those Y_j associated to graphlets containing i. Also, $g_\mathcal{S}(i) \leq \frac{1}{i} \sum_{u=1}^{n} g_\mathcal{S}(u)$ by the ordering of the nodes. Finally, $\sum_{u=1}^{n} g_\mathcal{S}(u) = ks = O(s)$, and thus $g_\mathcal{S}(i) = O(\frac{s}{i})$. Hence the second term in Equation 2 can be bounded as

$$\sum_{i=\ell+1}^{n} c_i^2 \leq \sum_{i=\ell+1}^{n} g_\mathcal{S}(i)^2 = \sum_{i=\ell+1}^{n} O\Big(\frac{s}{i}\Big)^2 = O\Big(\frac{s^2}{\ell}\Big). \tag{3}$$

The rest of the proof focuses on bounding $\sum_{i=1}^{\ell} c_i^2$. First of all, note that $\mathbb{E}[Y|X_1, \ldots, X_i] = \mathbb{E}[Y|X_1, \ldots, X_{i-1}]$ if the graphlet associated to Y does not contain i or does not contain one among $1, \ldots, i-1$. Therefore, c_i is bounded by the number of graphlets of $\mathcal{S}$ that contain both i and at least one of $1, \ldots, i-1$, and thus

$$c_i \leq \sum_{j=1}^{i-1} g_\mathcal{S}(i, j) \leq \sum_{j=1}^{i-1} g(i, j).$$

Assume now that $g(i, j) = O((g(i)+g(j))^{\frac{k-2}{k-1}})$, which we indeed prove later. By the ordering of nodes, $g(i)+g(j) = O(g/j)$, hence

$$c_i = O\Big(\sum_{j=1}^{i-1} (g/j)^{\frac{k-2}{k-1}}\Big) = O\big(g^{1-\frac{1}{k-1}} i^{\frac{1}{k-1}}\big).$$

Therefore

$$\sum_{i=1}^{\ell} c_i^2 = O\Big(\sum_{i=1}^{\ell} g^{2-\frac{2}{k-1}} i^{\frac{2}{k-1}}\Big) = O(g^{2-\frac{2}{k-1}} \ell^{1+\frac{2}{k-1}}) \quad (4)$$

and by setting $\ell = s^{1-\frac{1}{k}} g^{\frac{2}{k}-1}$ in both Equation 3 and 4 we obtain $\sum_{i=1}^{n} c_i^2 = O(s^{1+\frac{1}{k}} g^{1-\frac{2}{k}})$ as desired. We next prove:

LEMMA 9. *For any $u, v \in G$, $g(u,v) = O((g(u)+g(v))^{\frac{k-2}{k-1}})$.*

PROOF. For any $k \geq 1$ let $\mathcal{H}_k(u)$ denote the set of graphlets of size k of G that contain u (so $\mathcal{H}_1(u)$ contains only the trivial graphlet formed by u alone). For any graphlet occurrence H, let d_H be the sum of the degrees of its nodes in G; i.e., if $u_1, \ldots, u_k$ are the nodes of H then $d_H = \sum_{i=1}^{k} d_{u_i}$. To avoid ambiguities w.r.t. k, we use $g_k(u)$ instead of $g(u)$ to denote $|\mathcal{H}_k(u)|$. Note that $g_k(u) > 0$ for all k since G is connected by hypothesis, so we can safely employ Landau notation.

We start by proving that $g_k(u)$ is proportional to the sum of the degrees of the graphlets of size $k-1$ containing u:

$$\sum_{\mathcal{H}_{k-1}(u)} \frac{1}{k}\Big\lceil \frac{1}{k-1}(d_H - 2\tbinom{k-1}{2})\Big\rceil \leq g_k(u) \leq \sum_{\mathcal{H}_{k-1}(u)} d_H.$$

The upper bound follows immediately by noting that any element of $\mathcal{H}_k(u)$ can be obtained by adding to some $H \in \mathcal{H}_{k-1}(u)$ one of the neighbors of its nodes, and those neighbors are at most d_H. Consider now any $H \in \mathcal{H}_{k-1}(u)$. Since the arcs within H are at most $\binom{k-1}{2}$, and since G is connected, there must be at least $d_H - 2\binom{k-1}{2} > 0$ arcs between H and $G \setminus H$. These arcs then lead to at least $\lceil \frac{1}{k-1}(d_H - 2\binom{k-1}{2})\rceil$ distinct nodes, each of which can be added to obtain an element of $\mathcal{H}_k(u)$. Any element of $\mathcal{H}_k(u)$ can be obtained in this way, and from at most k elements of $\mathcal{H}_{k-1}(u)$. The lower bound then follows by summing $\lceil \frac{1}{k-1}(d_H - 2\binom{k-1}{2})\rceil$ over all $H \in \mathcal{H}_{k-1}(u)$ and dividing by k.

We can now prove the following crucial fact:

$$g_k(u) = \Omega\big(g_{k-1}(u)^{\frac{k-1}{k-2}}\big). \quad (5)$$

The proof is by an induction on k. The claim is obviously true for $k = 2$, so we prove it for $k+1$ assuming it holds for some $k \geq 2$. Since $g_{k+1}(u) = \Omega(\sum_{H' \in \mathcal{H}_k(u)} d_{H'})$, we will show the right-hand side is in $\Omega(g_k(u)^{\frac{k}{k-1}})$. Recall that from any $H \in \mathcal{H}_{k-1}(u)$ and its neighbors in G one can create $\lceil \frac{1}{k-1}(d_H - 2\binom{k-1}{2})\rceil = \Omega(d_H)$ graphlets of $\mathcal{H}_k(u)$; each such graphlet H' includes all the nodes of H, hence has degree $d_{H'} \geq d_H$, and may be obtained from at most k distinct graphlets H. Therefore:

$$\sum_{\mathcal{H}_k(u)} d_{H'} \geq \frac{1}{k} \sum_{\mathcal{H}_{k-1}(u)} \Omega(d_H) \cdot d_H = \Omega\Big(\sum_{\mathcal{H}_{k-1}(u)} d_H^2\Big).$$

By convexity and since $g_k(u) = \Omega(\sum_{\mathcal{H}_{k-1}(u)} d_H)$,

$$\sum_{\mathcal{H}_{k-1}(u)} d_H^2 \geq \frac{1}{g_{k-1}(u)}\Big(\sum_{\mathcal{H}_{k-1}(u)} d_H\Big)^2 = \Omega\Big(\frac{g_k(u)^2}{g_{k-1}(u)}\Big). \quad (6)$$

Now by the inductive hypothesis $g_{k-1}(u) = O(g_k(u)^{\frac{k-2}{k-1}})$, which used in the denominator of the right-hand side proves Equation 5.

We can now conclude the proof of Lemma 9. First of all note that any graphlet of size k containing both u and v is the union of (the sets of nodes of) two smaller graphlets: one of size h containing u (but possibly not v), and one of size $k-h$ containing v (but possibly not u), for some $h \in \{1, \ldots, k-1\}$. It follows that:

$$g(u,v) \leq \sum_{h=1}^{k-1} g_h(u) g_{k-h}(v).$$

Since k is a constant, we can thus choose an $h \in \{1, \ldots, k-1\}$ such that $g_h(u)g_{k-h}(v) \geq \Omega(g(u,v))$. If $h = 1$, since $g_1(u) = 1$ then $g_{k-1}(v) = \Omega(g(u,v))$, and $g_k(v) = \Omega((g(u,v))^{\frac{k-1}{k-2}})$ by Equation 5. Similarly, if $h = k-1$ we obtain $g_k(u) = \Omega((g(u,v))^{\frac{k-1}{k-2}})$.

Assume then $h \in \{2, \ldots, k-2\}$. If $g_h(u) = \Omega(g(u,v)^{\frac{h-1}{k-2}})$, by Equation 5 $g_k(u) = \Omega(g_h(u)^{\frac{k-1}{h-1}}) = \Omega(g(u,v)^{\frac{k-1}{k-2}})$. Otherwise $g_{k-h}(v) = \Omega(g(u,v)/g_h(u)) = \Omega(g(u,v)^{\frac{k-h-1}{k-2}})$, but then Equation 5 implies $g_k(v) = \Omega(g_{k-h}(v)^{\frac{k-1}{k-h-1}}) = \Omega(g(u,v)^{\frac{k-1}{k-2}})$. In any case $g(u)+g(v) = g_k(u)+g_k(v) = \Omega(g(u,v)^{\frac{k-1}{k-2}})$, which concludes the proof. □

4. GRAPHLETS VIA RANDOM WALKS

In this section we describe an algorithm that is based on a random walk on the graph. We start by describing the most natural random walk that can sample k-graphlets uniformly at random.

Recall that $\mathcal{V}_k(G)$ is the set of k-graphlets of G is connected. Consider then a new graph whose nodes are $\mathcal{V}_k(G)$. There is an edge between two graphlets iff the node set of one can be obtained by the node set of the other by removing one node and adding another. More precisely,

$$\mathcal{E}_k(G) = \{\{X, Y\} \mid X, Y \in \mathcal{V}_k(G) \text{ and } |X \cap Y| = k-1\}.$$

We let $\mathcal{G}_k(G)$ be the graph $\mathcal{G}_k(G) = (\mathcal{V}_k(G), \mathcal{E}_k(G))$. When k and G are clear from the context, we use the notation $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Observe that, since G is connected, the degrees of nodes in $\mathcal{G}$ range from 1 to $k \cdot \Delta(G) = k\Delta$. To make the random walk converge to the uniform distribution, we need to make the degrees of the nodes uniform. To do this, let $\mathcal{G}'$ be the following graph derived from $\mathcal{G}$: for each of its nodes $H \in \mathcal{V}$, add $(2k\Delta - \deg_{\mathcal{G}}(H))$ self-loops to H. This guarantees that all the nodes of $\mathcal{G}'$ have degree exactly $2k\Delta$. As an immediate consequence:

OBSERVATION 10. *The simple random walk on $\mathcal{G}'$ has a unique stationary distribution, which is uniform over its nodes.*

Therefore, after enough steps, the random walk will be on a node of $\mathcal{G}'$, i.e., a k-graphlet of G, chosen (almost) uniformly at random. The question is how many steps are needed, or in other words what is the *mixing time* [14] of the above random walk, i.e., the number of steps required to reach (within an ϵ-statistical error from) the stationary distribution. Let us start by recalling some standard notion.

Given a set $W \subseteq V$ of nodes, the *volume* of W is given by $\mathrm{vol}(W) = \sum_{v \in W} \deg(w)$. The *cut* induced by $W \subseteq V$ is equal to the number of edges that have exactly one endpoint in W, that is, $\mathrm{cut}(W) = |\{e : |e \cap W| = 1\}|$. The conductance of a set of nodes $W \subseteq V$ is defined as $\phi(W) = \frac{\mathrm{cut}(W)}{\mathrm{vol}(W)}$. The *conductance* of G is given by

$$\phi(G) = \min_{\substack{W \subseteq V \\ \mathrm{vol}(W) \leq |E|}} \phi(W).$$

Consider the uniform random walk on G, where at each node $v \in V$, the next node to visit is chosen uniformly at random from among the neighbors of v. Cheeger's inequality [7] implies that the mixing time of the walk is between $\Omega\left(\phi(G)^{-1}\right)$ and $O\left(\phi(G)^{-2} \log \frac{1}{\epsilon}\right)$.

A large conductance thus implies a small mixing time and vice versa.

Social graphs G have been empirically observed to have small mixing times [13]. A natural question then is: can we give small upper bounds on the mixing time of $\mathcal{G}'$ by using the fact that G has a small mixing time? We will show in Section 4.2 that, unfortunately, the answer to this question is negative in general: there are graphs G with very large (constant) conductance for which the corresponding $\mathcal{G}'$ has a tiny conductance. In fact, we will show a lower bound on the mixing time for the natural random walk on $\mathcal{G}$. Since $\mathcal{G}'$ is obtained by adding self-loops to $\mathcal{G}$, the lower bound on $\mathcal{G}$ directly translates to a lower bound on $\mathcal{G}'$. Before addressing these lower bounds, in the next section we give an upper bound on the mixing time of $\mathcal{G}'$ that may be of use in low-degree graphs.

4.1 An upper bound on the mixing time

LEMMA 11. *$\phi(\mathcal{G}'_k) \geq \Omega\left(n^{-1}\Delta^{-k}\right)$ and the mixing time of $\mathcal{G}'_k(G)$ can be upper bounded by $O\left(n^2\Delta^{2k}\right)$.*

PROOF. We first upper bound the number of k-graphlets of G, i.e., $|\mathcal{V}|$. Let us consider the set of "ordered" k-graphlets, i.e., the set $\mathcal{C} = \cup_{v\in\mathcal{V}}\mathcal{C}_v$, where $\mathcal{C}_v$ is the set of all the permutations of the k elements of v. Obviously, $|\mathcal{C}| = k! \cdot |\mathcal{V}|$, since each k-graphlet contains k distinct nodes.

For any given $v \in V$, we first count how many permutations $(v, v_1, \ldots, v_{k-1})$ are there in $\mathcal{C}$. Observe that v_1 has to be a neighbor of v, v_2 has to either be a neighbor of v or of v_1 (and has to be different from v and v_1) and, in general, v_i has to be different from and a neighbor of $v, v_1, \ldots, v_{i-2}$ or v_{i-1}. Therefore, v_i can be chosen in at most $i\Delta$ ways. It follows that, for a given v, the number of permutations $(v, v_1, \ldots, v_{k-1}) \in \mathcal{C}$ can be upper bounded by $\prod_{i=1}^{k-1}(i\Delta) = (k-1)! \cdot \Delta^{k-1}$. Since there are at most n ways of choosing $v \in V$, the cardinality of $\mathcal{C}$ satisfies: $|\mathcal{C}| \leq (k-1)! \cdot \Delta^{k-1} \cdot n$. It follows that $|\mathcal{V}| \leq (n/k) \cdot \Delta^{k-1}$.

Now, for $v \in \mathcal{V}$, $\deg(v) \leq k\Delta$. Therefore, the volume of any subset of nodes of $\mathcal{V}$ can be upper bounded by $\Delta k \cdot |\mathcal{V}| \leq n\Delta^k$. The connectedness of G implies the connectedness of $\mathcal{G}'_k$. Thus, any non-empty and proper subset of nodes of $\mathcal{G}'_k$ will have at least one edge in the cut. It follows that the conductance of $\mathcal{G}'_k$ is at least $\phi(\mathcal{G}'_k) \geq n^{-1} \cdot \Delta^{-k}$. The upper bound on the mixing time of $\mathcal{G}'_k$ then follows. □

4.2 Lower bounds on the mixing time

We next show a mixing time lower bound by exhibiting a graph G with large conductance, and such that $\mathcal{G}(G)$ has tiny conductance.

DEFINITION 12. *Let $k \in \mathbb{Z}^+$ be given. Let $\ell \in \mathbb{Z}^+$ be sufficiently large. Take ℓ disjoint paths of $2k$ nodes each; create two additional nodes a and b; for each path, add an edge between one of its endpoints and a, and an edge between its other endpoint and b. Let $G = (V, E)$ be the resulting graph.*

Observe that $|V| = 2\ell k + 2$; let $n = |V|$. We first observe (proof omitted) that the conductance of G is a constant.

LEMMA 13. *$\phi(G) = \Theta(1/k) = \Theta(1)$.*

We next prove that the conductance of $\mathcal{G}$ is tiny, which by Cheeger's inequality will imply that its mixing time is huge.

LEMMA 14. *$\phi(\mathcal{G}_k) \leq O\left(n^{2-k}\right)$ and hence its mixing time is at least $\Omega\left(n^{k-2}\right)$.*

PROOF. Consider the closed ball S of radius k centered at a. Observe that it is (i) disjoint and (ii) isomorphic to the closed ball T of radius k centered at b. Now, consider the set X of k-graphlets that contain only nodes in S, and the set Y of k-graphlets that contain only nodes in T.

By (ii), the subgraph that X induces on $\mathcal{G}_k$ will be isomorphic to the subgraph that Y induces on $\mathcal{G}_k$. Moreover, by (i), those two subgraphs will be disjoint. Thus, $\mathrm{vol}(X) \leq \mathrm{vol}(\mathcal{G}_k)/2 = \mathcal{E}_k/2$.

Moreover, $\mathrm{vol}(X) \geq \binom{(n-2)/(2k)}{k-1} \geq \Omega\left(n^{k-1}\right)$. Indeed, any subset of $k-1$ neighbors of a, together with a, forms a k-graphlet. Furthermore, $\mathrm{cut}(X) = \frac{n-2}{2k} \leq O(n)$.

Therefore, the conductance of the cut induced by X will be no more than $\phi(X) \leq O\left(n^{2-k}\right)$, and the conductance of $\mathcal{G}_k$ will then be upper bounded by the same quantity. □

Since $\mathcal{G}'$ is $\mathcal{G}$ with some extra self-loops, we have $\phi(\mathcal{G}'_k) \leq \phi(\mathcal{G}_k)$ and hence the mixing time of $\mathcal{G}'$ is $\Omega\left(n^{k-2}\right)$.

We observe that in time $O(n^k)$ one can just enumerate all the k-subsets of nodes of a graph of n nodes, check whether they form a k-graphlet and, if so, which graphlet do they form. As shown above, the random walk on $\mathcal{G}'_k$ has to run for at least $\Omega(n^{k-2})$ steps, to guarantee any statistical significance of the sampled graphlet.

Next, we show that not only the random walk does not converge in $o(n^{k-2})$ steps for some graphs with constant conductance, but in fact there are constant-conductance graphs such that $o(n^{k-1})$ steps of the random walk are not even enough to see any copy of a graphlet that occurs an overwhelming fraction (i.e., $1-o(1)$) of the times. This means we need $\Omega(n^{k-1})$ steps to see some occurrence of a graphlet appearing more than 99% of the times in the graph. We will consider a graph similar to the one in Definition 12.

DEFINITION 15. *Let $k \in \mathbb{Z}^+$ be given. Let $\ell \in \mathbb{Z}^+$ be sufficiently large. Take ℓ disjoint paths of $2k$ nodes each; create an additional node a and, for each path, add an edge between one of its endpoints and a. Construct a clique out of the ℓ other endpoints of the ℓ paths. Let $G = (V, E)$ be the resulting graph.*

As before, let $n = |V| = 2\ell k + 1$ and we can show:

LEMMA 16. *$\phi(G) = \Theta(1/k) = \Theta(1)$.*

We now observe that G contains a large fraction of k-cliques.

LEMMA 17. *The k-cliques are a $1 - o(1)$ fraction of the k-graphlets of G.*

PROOF. The number of k-cliques inside the clique of cardinality ℓ is equal to $\binom{\ell}{k} = \Theta\left(\ell^k\right) = \Theta\left(n^k\right)$. The number of graphlets that contain some node of the clique, and some node outside the clique is $\Theta(n^{k-1})$. The number of k-graphlets that contain a is no more than $\Theta\left(n^{k-1}\right)$. There are $\Theta(n)$ graphlets that do not contain nodes of the clique and a. Therefore the number of k-clique graphlets is a $1 - O(1/n)$ fraction of the total number of graphlets. □

LEMMA 18. *If we begin our random walk on $\mathcal{G}'_k(G)$ from any graphlet containing a and any $k-1$ of its neighbors, with high probability, we will require $\Omega(n^{k-1})$ steps to reach any graphlet that does not contain a. Therefore, with $o(n^{k-1})$ steps, with high probability the random walk will not visit any k-clique.*

PROOF SKETCH. Let us partition the set of graphlets that contain a into parts $P_1, \ldots, P_k$, where a graphlet is in part P_i if the maximum distance between one of its nodes and a is i. The starting graphlet is in P_1. We aim to show that it takes $\Omega\left(n^{k-1}\right)$ steps to reach P_k. Clearly, reaching some graphlet in P_k is necessary if

we are to reach some k-clique. Consider the walk between the P_i's. Observe that, if we are in P_i we can either remain there, or move to P_{i+1} (if $i < k$), or move back to P_{i-1} (if $i > 1$). Also observe that the probability of moving to P_{i+1} is no more than $\frac{k^2}{\ell} = O(n^{-1})$. Moreover, if $i > 1$, then the probability of reaching, in $O(k)$ steps, P_{i-1} from P_i is at least $1 - O(1/n)$. The time required to reach P_k, then, is $\Omega\left(n^{k-1}\right)$. □

It can be shown (proof omitted) that lower bounds similar to those of Lemma 18 also hold for other graphlet random walks that have been used in the literature.

5. EXPERIMENTS

This section presents three experiments. In the first, we use color coding algorithms to estimate the distribution of all graphlets of sizes 6, 7, and 8, on graphs ranging from a few thousand to a few million nodes. We confirm that the estimates appear to have low variance, in line with the concentration estimates of Section 3. In the second experiment, we measure the rate of convergence of the random walks to the uniform distribution. Such a rate drops very quickly with the size of the graph, suggesting that the bounds of Section 4.2 hold even if loosely. This also suggests that, perhaps surprisingly, random walk-based techniques might be of scarce help in the very task they were designed for. Finally, we compare the computational resource consumption of color coding versus random walks. As expected, the first is limited by space and the second by time constraints; yet color coding strikingly outperforms random walks on our largest graph for $k = 6$.

5.1 Setup

We ran our experiments on the largest connected component of the following real-world graphs.

name	nodes	edges	source
`WordAssoc`	10,617	63,788	LAW [3,4]
`Facebook`	63,392	816,886	MPI-SWS[3], [21]
`Yelp`	168,923	1,285,363	Dataset Chl.[4]
`Hollywood`	1,917,070	114,281,101	LAW [3,4]
`Orkut`	3,072,441	223,534,301	MPI-SWS[5], [15]
`LiveJournal`	5,363,186	49,514,271	LAW [3,4]
`Twitter`	41,652,230	117,185,083	LAW [3,4]

All graphs were treated as undirected. We note that `WordAssoc` is the only graph not representing a social network; we use it to investigate if networks from markedly different domains exhibit similarities.

Our code is written in Java and based on the WebGraph library[6]. It is publicly accessible at https://github.com/Steven--/graphlets. We ran it on a machine equipped with 240GiB of main memory and 32 Intel Xeon CPU cores at 2.50GHz with 25 MB of L3 cache, using Oracle's Java Virtual Machine (version 1.8.0).

5.2 Color coding: Going beyond five nodes

Using a carefully implemented version of CC1, we were able to estimate the distribution of graphlets of size 6 for all graphs except Twitter (by far the largest), and of size 7 for the three smallest graphs `WordAssoc`, `Facebook`, and `Yelp`.[7] Figure 1 and Figure 2 show the two distributions in logarithmic scale, graphlets sorted in nonincreasing order of average concentration in the graphs.

[6] http://webgraph.di.unimi.it/

[7] The same holds for CC1 except on `Orkut` where we estimated the distribution of graphlets up to size 5. The L_1 norm between the distributions obtained with CC1 and CC2 was never higher than 0.008.

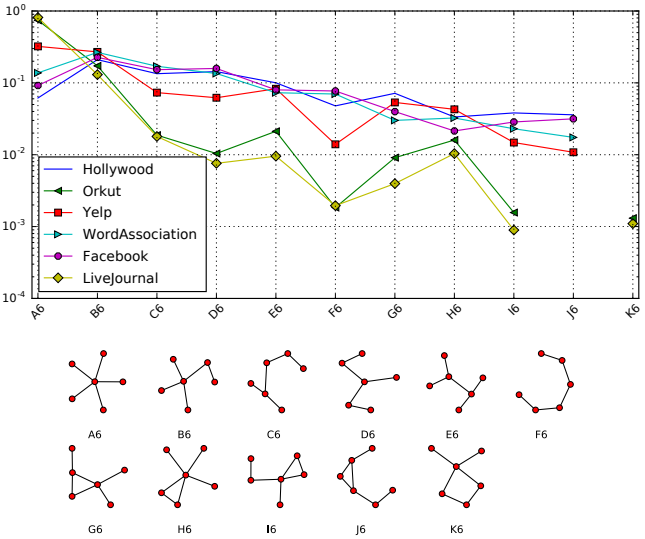

Figure 1: Distribution of graphlets of size 6.

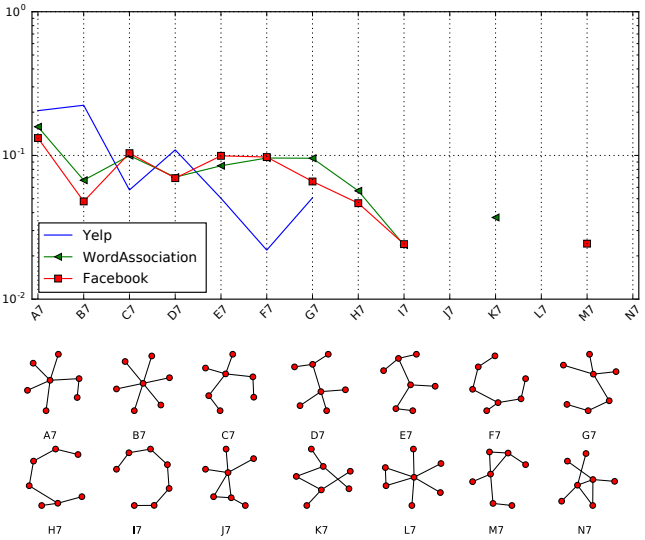

Figure 2: Distribution of graphlets of size 7.

Figures 1 and 2 tell some interesting facts. First, all the most frequent graphlets are trees. This might be rooted in social phenomena that we discuss in more detail in the next subsections. Second, our graphs are neatly partitioned in two families in the space of 6-graphlets distribution. On one side we have `Orkut` and `LiveJournal`, with highly skewed distributions that are strikingly similar; on the other we have the remaining graphs, with much flatter distributions that are again close (save perhaps `Yelp`). An intriguing question then is if the two families have been generated by radically different network formation processes. Finally, a surprising fact is that the distribution of `WordAssoc`—which is *not* a social graph—closely matches that of `Facebook`.

5.3 Convergence of random walks

In a second experiment, we measured the time the random walks took to converge to the stationary distribution. To this end, for

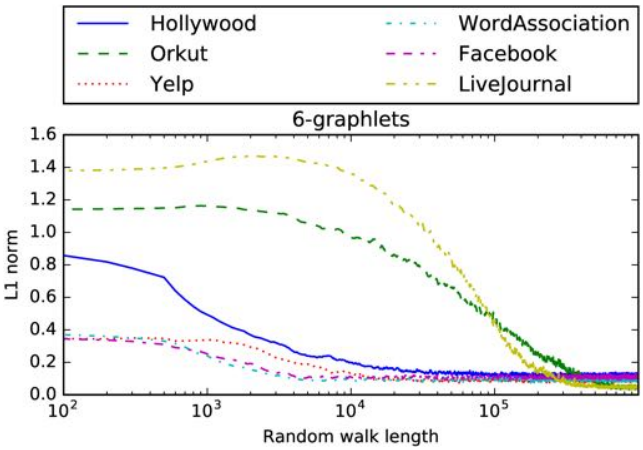

Figure 3: The convergence of sampling 6-graphlets with random walks (L_1 norm of the difference with ground truth).

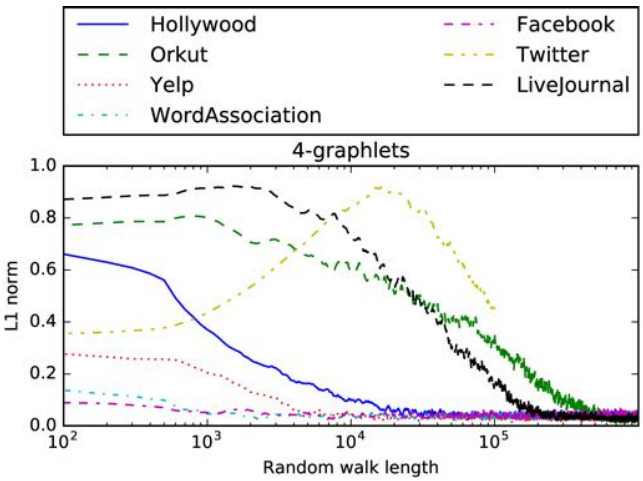

Figure 4: The convergence of sampling 4-graphlets with random walks (L_1 norm of the difference with ground truth).

WordAssoc and $k \leq 5$, Facebook and $k \leq 4$, and Yelp and $k = 3$ we computed the exact graphlet distribution through a recursive enumeration algorithm. In all other cases, we used as ground truth the distribution given by the average of ten color coding runs. We then sampled from the distribution of states reached by random walks after t steps, with t ranging from 1 to 1000000, using 1000 samples. Figure 3 shows the L_1 norm of the difference between the sampled distribution and the ground truth, as a function of t, for 6-graphlets (the situation for 5-graphlets is analogous) while Figure 4 refers to 4-graphlets. It is interesting to notice how, even for 4-graphlets, 100000 steps are not enough for the random walk to mix on Twitter.

5.4 Space vs. time

Comparing the computational resources of the two methods when running the experiments gives us with interesting insights.

On the one hand, consider time. We compared the time needed by one execution of CC1 with the time needed by the random walk to reach the same L_1 norm (see above). In both algorithms we sampled 1000 graphlets. We do not consider runs that took less than 5 seconds, for a series of reasons (notably the JVM warm-up time). Among the runs taking 5 or more seconds, color coding and random walks were almost always comparable with a factor ≤ 4 between their running times; however, this could easily be dwarfed by speedups obtained through code optimizations or hardware modifications[8]. There were however notable exceptions. Random walks outperformed color coding on Facebook by a factor of nearly 5; both terminated in less than a minute for all k. Interestingly, the situation drastically reversed on LiveJournal and Orkut: color coding was more than 10 times faster than random walks—for $k = 6$ and $k = 5$, respectively, the running times were 1.8 hours versus less than 8 minutes and 1.4 hours versus less than 6 minutes (see Figure 5).

On the other hand, consider space. We note that color coding never exceeded the allocated time budget, but it exceeded the available main memory (240 GiB); this was never the case for random walks. Notwithstanding, color coding provided us with graphlet distributions that extended the current state of the art in terms of graph and graphlet sizes. After all, its severe dependency on k might not be a practical limiting factor yet.

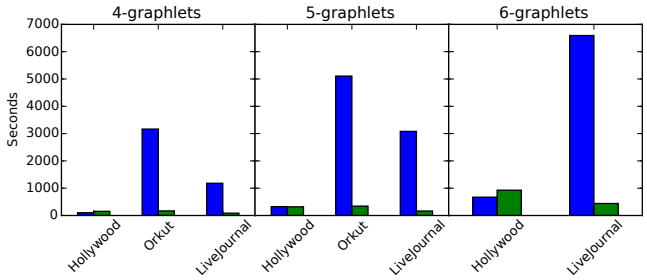

Figure 5: Comparison of the time needed for the CC1 and the random walk algorithms to sample 1000 k-graphlets. For $k = 3$ and for the graphs smaller than the one shown, the times were always below 90 seconds and they are not reported. Notice that the time of CC1 also includes the building phase of the algorithm.

5.5 Some observations

We now comment on some of the findings on motifs enabled by our algorithms. In all our graphs and for all k's (Figures 1, 2), the most frequent graphlet is either K_{k-1} (a star) or the graphlet obtained by attaching an extra node to one leaf of K_{k-2}; we denote the latter by K^+_{k-2}. Moreover, if we take the average of the graphlet frequencies in the various graphs, we see that the six most frequent 6-graphlets, and the nine most frequent 7-graphlets are trees. In addition, many of these tree-shaped graphlets have a single *branching* node, i.e., a single node with degree more than two. This suggests that many of the most frequent graphlets have a center (the branching node) mapped to a high-degree node (*hub*) of the graph.

We also observe that LiveJournal contains many more copies of K_{k-1} than of K^+_{k-2}. A possible explanation is that the readers of the most frequently read blogs will tend not to know each other and since the LiveJournal graph was made undirected, the neighbors of the high-degree nodes will tend to not induce many edges. A similarly strong imbalance can be seen in the Orkut graph. This might be because the Orkut graph has a very large maximum degree ($\sim$ 32K)[9]. Friends of high-degree celebrities tend to have few friendship edges between themselves, and this can explain the preponderance of the star in the Orkut graph.

[8]For instance, color coding heavily accesses vast memory regions in a random fashion, whereas random walks are rather limited by the CPU and its data cache.

[9]Orkut did not impose an upper bound on the number of friends a user could have.

The Facebook graph (containing only the users from the New Orleans area) imposes an upper bound on the number of friends of a user. Therefore, the likelihood that two neighbors of a node are actually friends is larger, and the ego-network of most nodes will tend to have a significant number of edges. This clearly decreases the fraction of induced K_{k-1}'s that can be found in this graph.

The Hollywood graph is the union of cliques, since all the actors that starred in a movie will be pairwise friends. Therefore, this graph does not contain many induced K_{k-1}'s.

6. RELATED WORK

The naive algorithm for counting the exact number of occurrences of all graphlets of size k in an n-node graph by enumeration takes $O(k^2 n^k)$ time. Faster exact algorithms are known [8, 25], but their complexity remains $n^{\Theta(k)}$ and are infeasible in practice already for moderate values of n and k. Indeed, the problem is #W[1]-hard and thus unlikely to admit an $f(k)n^{O(1)}$-time algorithm [10].

Counting graphlets in practice is thus always performed using approximate algorithms or heuristics. One heuristic approach is *path sampling*, a technique introduced in [12], which consists of sampling a path of k nodes uniformly at random from G and check the graphlet they induce; all occurrences of graphlets containing a path of length k will then have roughly the same probability of coming up. This can be done efficiently for $k = 4$ [12], and can be adapted to $k = 5$ by using a set of sampling strategies for trees on five nodes [23, 24]. However, for $k > 5$ this method becomes overly intricate and can significantly suffer from rejection (the k nodes sampled may not be distinct). In contrast, we aim at sampling graphlets of size k larger than four and five through approaches that give provable guarantees.

The first random walk-based algorithm, GUISE, was introduced in [2] and allowed the authors to collect, in just a few minutes, samples of graphlets of size $k = 4$ and 5 in graphs with up to 4 million nodes—a significant advancement of the state of the art at the time. Further generalizations and refinings of this technique followed [6, 9, 22], confirming its prowess at least for sampling graphlets of size $k \leq 5$ in graphs of a few million nodes (and $k = 6$ on one small graph). Also, a handful of theoretical results have shed light on the number of samples needed for the walk to converge to stationarity, and thus on the theoretical efficiency of the method. However, no general result has been given that is a function of G and k, the two inputs to the problem. Obtaining such a result is part of the goals of our work.

The attractive bounds offered by color coding has made it possible to push the task of estimating subgraph counts in the realm of graphs with millions of nodes. A first distributed algorithm based on color coding, PARSE [26], was used to count each of 7 types of subgraphs of size from 4 to 10 in graphs with up to 20 million nodes; the algorithms works for subgraphs that can be disconnected by cutting a single edge. A subsequent distributed scalable implementation of color coding, SCALA [16], allowed the authors to count subgraphs with size up to $k = 7$ on networks with 1—2M nodes; again, the considered subgraphs are limited to non-induced paths and trees. The most recent effort towards scaling color coding is [5]: using a distributed algorithm, the authors estimate the occurrences of ten different subgraphs of treewidth 2 and size up to $k = 10$ nodes, in graphs of up to 2M nodes. While these existing and encouraging results make clear that color coding is a promising approach, they leave wide open the important question of estimating the distribution of *induced* subgraphs, i.e., graphlets. In this paper, we show how color coding fits the purpose—with almost no overhead.

Finally, we remark that currently no comparison, either theoretical or experimental, exists between the two main subgraph counting techniques—random walks and color coding. Such a comparison is one of the goals of our work.

7. CONCLUSIONS

In this paper we studied MC and CC as powerful algorithmic methods to solve the problem of graphlet counting in massive graphs. Our mixing time analysis on MC cautions its blind use on real graphs, if statistical accuracy is paramount. However, on some graphs, it performs as well as CC, suggesting that there are structural properties that might be playing a role in the mixing rate. Investigating this both theoretically and empirically are interesting research directions. For CC, it will be interesting to see if the dynamic program table can somehow be compressed without sacrificing statistical guarantees much, which would make the method applicable for even larger k's. However, such an endeavor appears very challenging.

8. REFERENCES

[1] N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[2] M. A. Bhuiyan, M. Rahman, M. Rahman, and M. Al Hasan. Guise: Uniform sampling of graphlets for large graph analysis. In *ICDM*, pages 91–100, 2012.

[3] P. Boldi, M. Rosa, M. Santini, and S. Vigna. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *WWW*, pages 587–596, 2011.

[4] P. Boldi and S. Vigna. The WebGraph framework I: Compression techniques. In *WWW*, pages 595–601, 2004.

[5] V. T. Chakaravarthy, M. Kapralov, P. Murali, F. Petrini, X. Que, Y. Sabharwal, and B. Schieber. Subgraph counting: Color coding beyond trees. In *IPDPS*, pages 2–11, 2016.

[6] X. Chen, Y. Li, P. Wang, and J. C. S. Lui. A general framework for estimating graphlet statistics via random walk. *CoRR*, abs/1603.07504, 2016.

[7] F. Chung. Four proofs for the Cheeger inequality and graph partition algorithms. In *ICCM*, 2007.

[8] P. Floderus, M. Kowaluk, A. Lingas, and E.-M. Lundell. Detecting and counting small pattern graphs. *SIAM J. Discrete Math.*, 29(3):1322–1339, 2015.

[9] G. Han and H. Sethu. Waddling random walk: Fast and accurate sampling of motif statistics in large graphs. *CoRR*, abs/1605.09776, 2016.

[10] M. Jerrum and K. Meeks. The parameterised complexity of counting connected subgraphs and graph motifs. *J. Comput. Syst. Sci.*, 81(4):702–716, 2015.

[11] M. Jha, C. Seshadhri, and A. Pinar. A space efficient streaming algorithm for triangle counting using the birthday paradox. In *KDD*, pages 589–597, 2013.

[12] 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.

[13] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW*, pages 695–704, 2008.

[14] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009.

[15] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and Analysis of Online

Social Networks. In *ACM/Usenix IMC*, 2007.

[16] G. M. Slota and K. Madduri. Fast approximate subgraph counting and enumeration. In *ICPP*, pages 210–219, 2013.

[17] S. Suri and S. Vassilvitskii. Counting triangles and the curse of the last reducer. In *WWW*, pages 607–614, 2011.

[18] N. H. Tran, K. P. Choi, and L. Zhang. Counting motifs in the human interactome. *Nature Communications*, 4(2241), 2013.

[19] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos. DOULION: counting triangles in massive graphs with a coin. In *KDD*, pages 837–846, 2009.

[20] W. T. Tutte. *Graph Theory*. Cambridge University Press, 2001.

[21] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in Facebook. In *WOSN*, pages 37–42, 2009.

[22] P. Wang, J. C. S. Lui, B. Ribeiro, D. Towsley, J. Zhao, and X. Guan. Efficiently estimating motif statistics of large networks. *TKDD*, 9(2):8:1–8:27, 2014.

[23] P. Wang, J. Tao, J. Zhao, and X. Guan. Moss: A scalable tool for efficiently sampling and counting 4- and 5-node graphlets. *CoRR*, abs/1509.08089, 2015.

[24] P. Wang, X. Zhang, Z. Li, J. Cheng, J. C. S. Lui, D. Towsley, J. Zhao, J. Tao, and X. Guan. A fast sampling method of exploring graphlet degrees of large directed and undirected graphs. *ArXiv e-prints*, 2016.

[25] V. V. Williams and R. Williams. Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013.

[26] Z. Zhao, M. Khan, V. S. A. Kumar, and M. V. Marathe. Subgraph enumeration in large social contact networks using parallel color coding and streaming. In *ICPP*, pages 594–603, 2010.