`PRESTO`: Simple and Scalable Sampling Techniques for the Rigorous Approximation of Temporal Motif Counts[*]

Ilie Sarpe[†] Fabio Vandin[†]

Abstract

The identification and counting of small graph patterns, called *network motifs*, is a fundamental primitive in the analysis of networks, with application in various domains, from social networks to neuroscience. Several techniques have been designed to count the occurrences of motifs in static networks, with recent work focusing on the computational challenges provided by large networks. Modern networked datasets contain rich information, such as the time at which the events modeled by the networks edges happened, which can provide useful insights into the process modeled by the network. The analysis of motifs in temporal networks, called *temporal motifs*, is becoming an important component in the analysis of modern networked datasets. Several methods have been recently designed to count the number of instances of temporal motifs in temporal networks, which is even more challenging than its counterpart for static networks. Such methods are either exact, and not applicable to large networks, or approximate, but provide only weak guarantees on the estimates they produce and do not scale to very large networks. In this work we present an efficient and scalable algorithm to obtain rigorous approximations of the count of temporal motifs. Our algorithm is based on a simple but effective sampling approach, which renders our algorithm practical for very large datasets. Our extensive experimental evaluation shows that our algorithm provides estimates of temporal motif counts which are more accurate than the state-of-the-art sampling algorithms, with significantly lower running time than exact approaches, enabling the study of temporal motifs, of size larger than the ones considered in previous works, on billion edges networks.

1 Introduction

The identification of patterns is a ubiquitous problem in data mining [8] and is extremely important for networked data, where the identification of small, connected subgraphs, usually called *network motifs* [18] or *graphlets* [23], have been used to study and characterize networks from various domains, including biology [16], neuroscience [3], social networks [28], and the study of complex systems in general [17]. Network motifs have been used as building blocks for various tasks in the analyses of networks across such domains, including anomaly detection [25] and clustering [4].

A fundamental problem in the analysis of network motifs is the counting problem [5, 2], which requires to output the number of instances of the given topology defining the motif. This challenging computational problem has been extensively studied, with several techniques designed to count the number of occurrences of simple motifs, such as triangles [26, 21, 24] or sparse motifs [7].

Most recent work has focused on providing techniques for the analysis of *large* networks, which have become the standard in most applications. However, in addition to a significant increase in size, modern networks also feature a richer structure, in terms of the type of information that is available for their vertices and edges [6]. A type of information that has drawn significant attention in recent years is provided by the temporal dimension [11, 12]. In several applications edges are supplemented with *timestamps* describing the time at which an event, modeled by an edge, occurred: for example, in the analysis of spreading processes in epidemics, nodes are individuals, an edge represents a physical interaction between two individuals, and the timestamp represents the time at which the interaction was recorded [22].

When studying motifs in temporal networks, one is usually interested in occurrences of a given topology whose edge timestamps all appear in a small time span [11, 20]. Discarding the temporal information of the network, i.e. ignoring the timestamps, may lead to incorrect characterization of the system of interest, while the analysis of temporal networks can provide insights that are not revealed when the temporal information is not accounted for [11]. For example, in a temporal network, a triangle $x \to y \to z \to x$ represents some feedback process on the information originated from x

[*]This work was supported, in part, by MIUR of Italy, under PRIN Project n. 20174LF3T8 AHeAD and grant L. 232 (Dipartimenti di Eccellenza), and by the University of Padova under projects "STARS 2017: Algorithms for Inferential Data Mining" and "SID 2020: RATED-X".

[†]Dept. of Information Engineering, University of Padova, Italy. Email: `{sarpeilie,vandinfa}@dei.unipd.it`

only if the edges occur at increasing timestamps (and the triangle occurs in a small amount of time). This information is revealed only by considering the timestamps, while by restricting to the static network (i.e., discarding edges timestamps) we may, often incorrectly, conclude that initial information starting from x always affects such sequence of events. Motifs that capture temporal interactions, such as the ones we consider, can provide more useful information than static motifs, as shown in several applications, including network classification [27] and in the identification of mixing services from bitcoin networks [30]. Furthermore, while on static networks motifs with high counts are associated with important properties of the dataset (e.g., its domain), the temporal information provides additional insights on the network and the nature of frequently appearing motifs, such as, for example, the presence of bursty activities [32]. Unfortunately, current techniques do not enable the analysis of large temporal networks, preventing researchers from studying temporal motifs in complex systems from many areas.

The problem of counting motifs in temporal networks is even more challenging than its counterpart for static networks, since there are motifs for which the problem is NP-hard for temporal networks while it is efficiently solvable for static networks [13]. Current approaches to count motifs in temporal networks are either *exact* [20, 15], and cannot be employed for very large networks, or *approximate* [13, 29], but provide only rather weak guarantees on the quality of the estimates they return. In addition, even approximate approaches do not scale to billion edges networks.

In this work we focus on the problem of counting motifs in temporal networks. Our goal is to obtain an efficiently computable approximation to the count of motifs of *any* topology while providing rigorous guarantees on the quality of the result.

Our contributions This work provides the following contributions.

- We present `PRESTO`, an algorithm to approximate the count of motifs in temporal networks, which provides rigorous (probabilistic) guarantees on the quality of the output. We present two variants of `PRESTO`, both based on a common approach that counts motifs within randomly sampled temporal windows. Both variants allow to analyze billion edges datasets providing sharp estimates. `PRESTO` features several useful properties, including: i) it has only one easy to interpret parameter, c, defining the length of the temporal windows for the samples; ii) it can approximate the count of *any* motif topology; iii) it is easily parallelizable, with an almost linear speed-up with the available processors.
- We provide tight and efficiently computable bounds to the number of samples required by our algorithms to achieve (multiplicative) approximation error $\leq \varepsilon$ with probability $\geq 1-\eta$, for given $\varepsilon > 0$ and $\eta \in (0,1)$. Our bounds are obtained through the application of advanced concentration results (i.e., Bennett's inequality) for the sum of independent random variables.
- We perform an extensive experimental evaluation on real datasets, including a dataset with more than 2.3 billion edges, never examined before. The results show that on large datasets our algorithm `PRESTO` significantly improves over the state-of-the-art sampling algorithms in terms of quality of the estimates while requiring a small amount of memory.

2 Preliminaries

In this section we introduce the basic definitions used throughout this work. We start by formally defining temporal networks.

Definition 2.1. *A* temporal network *is a pair $T = (V, E)$ where, $V = \{v_1, \ldots, v_n\}$ and $E = \{(x, y, t) : x, y \in V, x \neq y, t \in \mathbb{R}^+\}$ with $|V| = n$ and $|E| = m$.*

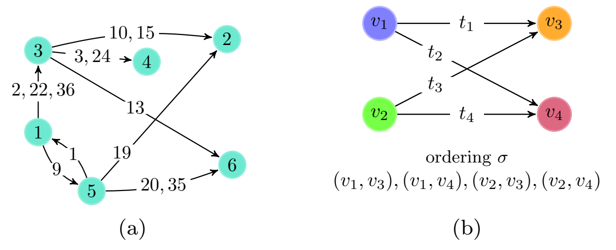
Given $(x, y, t) \in E$, we say that t is the *timestamp* of the edge (x, y). For simplicity in our presentation we assume the timestamps to be unique, which is without loss of generality since in practice our algorithms also handle non-unique timestamps. We also assume the edges to be sorted by increasing timestamps, that is $t_1 < \cdots < t_m$. Given an interval or *window* $[t_B, t_E] \subseteq \mathbb{R}$ we will denote $|t_E - t_B|$ as its *length*.

We are interested in *temporal motifs*[1], which are small, connected subgraphs whose edge timestamps satisfy some constraints. In particular, we consider the following definition introduced in [20].

Definition 2.2. *A k-node ℓ-edge* temporal motif *M is a pair $M = (\mathcal{K}, \sigma)$ where $\mathcal{K} = (V_\mathcal{K}, E_\mathcal{K})$ is a directed and* weakly connected *multigraph where $V_\mathcal{K} = \{v_1, \ldots, v_k\}$, $E_\mathcal{K} = \{(x, y) : x, y \in V_\mathcal{K}, x \neq y\}$ s.t. $|V_\mathcal{K}| = k$ and $|E_\mathcal{K}| = \ell$ and σ is an ordering of $E_\mathcal{K}$.*

Note that a k-node ℓ-edge temporal motif $M = (\mathcal{K}, \sigma)$ is also identified by the sequence $(x_1, y_1), \ldots,$

[1] In static networks, the term *graphlet* [31] is sometimes used, with *motifs* denoting statistically significant graphlets. We use the term motif in accordance with previous work on temporal networks [20, 13, 29].



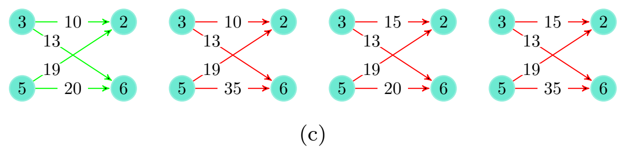


Figure 1: (1a): example of temporal network T with $n = 6$ nodes and $m = 13$ edges. (1b): a temporal motif, known as Bi-Fan [13]. (1c): sequences of edges of T that map topologically on the Bi-Fan motif, i.e., in terms of static graph isomorphism. For $\delta = 10$ only the green sequence is a δ-instance of the Bi-Fan, since the timestamps respect σ and $t'_\ell - t'_1 = 20 - 10 \leq \delta$. The red sequences are not δ-instances, since they do not respect such constraint or do not respect the order in σ.

(x_ℓ, y_ℓ) of edges ordered according to σ. Given a k-node ℓ-edge temporal motif M, k and ℓ are determined by $V_\mathcal{K}$ and $E_\mathcal{K}$. We will therefore use the term *temporal motif*, or simply *motif*, when k and ℓ are clear from context.

Given a temporal motif M, we are interested in counting how many times it appears within a time *duration* of δ, as captured by the following definition.

Definition 2.3. *Given a temporal network $T = (V, E)$ and $\delta \in \mathbb{R}^+$, a* time ordered *sequence $S = (x'_1, y'_1, t'_1), \ldots, (x'_\ell, y'_\ell, t'_\ell)$ of ℓ unique temporal edges from T is a δ-instance of the temporal motif $M = (x_1, y_1), \ldots, (x_\ell, y_\ell)$ if:*

1. *there exists a bijection f on the vertices such that $f(x'_i) = x_i$ and $f(y'_i) = y_i$, $i = 1, \ldots, \ell$; and*
2. *the edges of S occur within δ time, i.e., $t'_\ell - t'_1 \leq \delta$.*

Note that in a δ-instance of the temporal motif $M = (\mathcal{K}, \sigma)$ the edge timestamps must be sorted according to the ordering σ. See Figure 1 for an example.

Let $\mathcal{U}(M, \delta) = \{u : u \text{ is a } \delta\text{-instance of } M\}$ be the *set of (all) δ-instances* of the motif M in T, denoted only with $\mathcal{U}$ when M and δ are clear from the context. Given a δ-instance $u \in \mathcal{U}(M, \delta)$, we denote the timestamps of its first and last edge with t_1^u and t_ℓ^u, respectively. The *count* of M is $C_M(\delta) = |\mathcal{U}(M, \delta)|$, denoted with C_M when δ is clear from the context. We are interested in solving the following problem.

Motif counting problem. Given a temporal network T, a temporal motif $M = (\mathcal{K}, \sigma)$, and $\delta \in \mathbb{R}^+$, compute the count $C_M(\delta)$ of δ-instances of motif M in the temporal network T.

Solving the motif counting problem *exactly* may be infeasible for large networks, since even determining whether a temporal network contains a simple *star motif* is NP-hard [13]. State-of-the-art exact techniques [15, 20] require exponential time and memory in the number of edges of the temporal network, which renders them impractical for large temporal networks. We are therefore interested in obtaining efficiently computable approximations of motif counts, as follows.

Motif approximation problem. Given a temporal network T, a temporal motif $M = (\mathcal{K}, \sigma)$, $\delta \in \mathbb{R}^+$, $\varepsilon \in \mathbb{R}_0^+, \eta \in (0, 1)$ compute C'_M such that $\mathbb{P}(|C'_M - C_M(\delta)| \geq \varepsilon C_M(\delta)) \leq \eta$, i.e., C'_M is a *relative ε-approximation* to $C_M(\delta)$ with probability at least $1 - \eta$.

We call an algorithm that provides such guarantees an *(ε, η)-approximation algorithm.*

2.1 Related Work

Various definitions of temporal networks and motifs have been proposed in the literature; we refer the interested reader to [10, 12, 14]. Here we focus on those works that adopted the same definitions used in this work.

The definition of temporal motif we adopt was first proposed by Paranjape et al. [20], which provided efficient exact algorithms to solve the motif counting problem for specific motifs. Such algorithms are efficient only for specific motif topologies and do not scale to very large datasets. An algorithm for the counting problem on general motifs has been introduced by Mackey et al. [15]. Their algorithm is the first exact technique allowing the user to enumerate all δ-instances $u \in \mathcal{U}$ without any constraint on the motif's topology. The major back-draw of such algorithm is that it may be impractical even for moderately-sized networks, due to its exponential time complexity and memory requirements.

Liu et al. [13] proposed the first sampling algorithm for the temporal motif counting problem. The main strategy of [13] is to partition the time interval containing all the edges of the network into *non overlapping* and contiguous windows of length $c\delta$ (i.e., a grid-like partition), for some $c > 1$. The partition is then randomly shifted (i.e., the starting point of the first window may not coincide with the smallest timestamp of the network). The edges in each partition constitute the candidate samples to be analyzed using an exact algorithm. An *importance sampling* scheme is used to sample (approximately) r windows among the candidates, with a window being selected with probability proportional to the fraction of edges it contains. The estimate for each sampled window is obtained by weighting each δ-instance in the window, and the estimates are averaged across windows. This procedure is repeated b times to reduce the variance of the estimate. While interesting, this partition-based approach prevents such algorithm to provide (ε, η)-guarantees (see Sec. 3.1).

Recently Wang et al. [29] proposed an (ε, η)-approximation algorithm for the motif counting problem. Their approach selects each edge in T with a user-provided probability p. Then for each selected edge $e = (x, y, t)$, the algorithm collects the edges with timestamps in the edge-centered window $[t-\delta, t+\delta]$, of length 2δ, computes on these edges all the δ-instances $u \in \mathcal{U}$ containing e, weights the instances, and combines the weights to obtain the final estimate. From the theoretical point of view, the main drawback of this approach is that in order to achieve the desired guarantees one has to set $p \geq 1/(1 + \eta\varepsilon^2)$, resulting in high values of p (i.e., almost all edges are selected) for reasonable values of η and ε (e.g., $p > 0.97$ for $\eta = 0.1$ and $\varepsilon = 0.5$). In addition, such approach is impractical on very large datasets, mainly due its huge memory requirements, and does not provide accurate estimates (see Sec. 4).

3 PRESTO: Approximating Temporal Motif Counts with Uniform Sampling

We now describe and analyze our algorithm PRESTO (apPRoximating tEmporal motifS counTs with unifOrm sampling) for the motif counting problem. We start by describing, in Sec. 3.1, the common strategy underlying our algorithms. We then briefly highlight the differences between PRESTO and the existing sampling algorithms for the counting problem. In Sec. 3.2 and Sec. 3.3 we present and analyze two variants of the common strategy introduced in Sec. 3.1. We conclude with the analysis of PRESTO's complexity in Sec. 3.4.

3.1 PRESTO: General Approach The general strategy of PRESTO is presented in Algorithm 1. Given a network T, PRESTO collects s samples, where each sample is obtained by gathering all edges $e \in E$ with timestamp in a *small random* window $[t_r, t_r + c\delta]$ of length $c\delta$, using SampleWindowStart(E) (Lines 2-5) to select t_r (i.e., the starting time point of the window) through a uniform random sampling approach. For each sample, an exact algorithm is then used to extract all the δ-instances of motif M in such sample (Line 6). The weight of each δ-instance extracted is computed with the call ComputeWeight(u) (Line 8), and the sum of all such weights constitutes the estimate provided by the sample (Line 9). The weights account for the probability of sampling the δ-instances in the sample, making the final estimate *unbiased*. The final estimate produced in output is the average of the samples' estimates (Lines 10-11). Note that the for cycle of Line 1 is trivially parallelizable. The two variants of PRESTO we will present differ in the way i) SampleWindowStart(E) and ii) ComputeWeight(u) are defined.

Algorithm 1: PRESTO

```
   Input: Temporal Network $T = (V, E)$, Motif
          $M$, Motif Duration $\delta > 0$, $s > 0$, $c > 1$
   Output: Estimate $C'_M$ of $C_M$
1  for $i \leftarrow 1$ to $s$ do
2      $t_r \leftarrow$ SampleWindowStart($E$);
3      $E_i \leftarrow \{(x, y, t) \in E : (t \geq t_r) \wedge (t \leq t_r + c\delta)\}$;
4      $V_i \leftarrow \{x : ((x, y, t) \in E_i) \vee ((y, x, t) \in E_i)\}$;
5      $T_i \leftarrow (V_i, E_i)$; $X_i \leftarrow 0$;
6      $\mathcal{S}_i \leftarrow \{u : u$ is $\delta$-instance of $M$ in $T_i\}$;
7      foreach $u \in \mathcal{S}_i$ do
8          $w(u) \leftarrow$ ComputeWeight($u$);
9          $X_i \leftarrow X_i + w(u)$;
10 $C'_M \leftarrow \frac{1}{s}\sum_{i=1}^{s} X_i$;
11 return $C'_M$;
```

Differently from [13], our algorithm PRESTO does not partition the edges of T in non overlapping windows, and relies instead on *uniform sampling*. We recall (see Sec. 2.1) that in [13], after computing all the non overlapping intervals defining the candidate samples, there may be several δ-instances $u \in \mathcal{U}$ that cannot be sampled (i.e., all δ-instances having t_1^u in window j and t_ℓ^u in window $j+1$). This significantly differs from PRESTO, which instead samples at each iteration a small random window from $[t_1, t_m]$ without restricting the candidate windows, allowing to sample *any* δ-instance $u \in \mathcal{U}(M, \delta)$ at each iteration. This enables us to provide stronger guarantees on the quality of the output, since each δ-instance has a non-zero probability of being sampled at each step, leading to (ε, η)-approximation guarantees.

Differently from the work of Wang et al. [29] PRESTO samples temporal windows of length $c\delta$ and does not follow the edge-centric approach (i.e., sampling temporal edges with a user-provided probability p) of [29]. In addition, the approach of [29] collects edges in temporal-windows of length 2δ (see Sec. 2.1), while in PRESTO the window size is controlled by the parameter c, which, when fixed to be < 2, leads to much more scalability than [29] while requiring less memory (see Sec. 4).

3.2 PRESTO-A: Sampling among All Windows In this section we present and analyze PRESTO-A, our first (ε, η)-approximation algorithm obtained by specifying i) how the starting point t_r of the temporal window defining a sample T_i is chosen (function `SampleWindowStart`(E) in Line 2) and ii) how the weight $w(u)$ of a δ-instance u in a sample is computed (`ComputeWeight`(u), Line 8).

The starting point t_r of sample T_i is sampled uniformly at random in the interval $[t_\ell - c\delta, t_{m-\ell}] \subseteq \mathbb{R}$, where we recall $\ell = |E_\mathcal{K}|$ and $m = |E|$. Regarding the choice of the weight $w(u)$ for each instances $u \in \mathcal{S}_i$, PRESTO-A considers $w(u) = 1/p_u$, with p_u being the probability of u to be in $\mathcal{S}_i$, that is $p_u = r_u/\Delta_{T,1}$, where $\Delta_{T,1} = t_{m-\ell} - t_\ell + c\delta$ is the total length of the interval from which t_r is sampled (recall that we choose t_r from $[t_\ell - c\delta, t_{m-\ell}]$), and $r_u = c\delta - (t^u_\ell - t^u_1)$ is the length of the interval in which t_r must be chosen for u to be in $\mathcal{S}_i, i = 1, \dots, s$.

We now present the theoretical guarantees of PRESTO-A and give efficiently computable bounds for the sample size s needed for the (ε, η)-approximation to hold. (All the missing proofs are in the extended version [34]). Recall the definition of $\mathcal{U}(M, \delta)$, which is the set of δ-instances of M in T. Let u be an arbitrary δ-instance of motif M, and let T_i be an arbitrary sample obtained by PRESTO-A at its i-th iteration. We define the following set of indicator random variables, for $u \in \mathcal{U}$ and for $i = 1, \dots, s$: $X^i_u = 1$ if $u \in \mathcal{S}_i$, 0 otherwise. Each variable $X^i_u, i = 1, \dots, s, u \in \mathcal{U}$ is a Bernoulli random variable with $\mathbb{P}(X^i_u = 1) = \mathbb{P}(u \in \mathcal{S}_i) = \frac{r_u}{\Delta_{T,1}} = p_u$. Therefore, for each variable X^i_u, it holds $\mathbb{E}[X^i_u] = p_u$. Thus, for each $i = 1, \dots, s$, iteration i provides an estimate X_i of C_M, which is the random variable: $X_i = \sum_{u \in \mathcal{U}} \frac{1}{p_u} X^i_u$. We therefore have the following result.

Lemma 3.1. *For each $i = 1, \dots, s$, X_i and $C'_M = \frac{1}{s}\sum_{i=1}^{s} X_i$ are unbiased estimators for C_M, that is $\mathbb{E}[X_i] = \mathbb{E}[C'_M] = C_M$.*

The following result provides a bound on the variance of the estimate of C_M provided by C'_M.

Lemma 3.2. *For PRESTO-A it holds*

$$\mathrm{Var}\left(C'_M\right) = \mathrm{Var}\left(\frac{1}{s}\sum_{i=1}^{s} X_i\right) \le \frac{C_M^2}{s}\left(\frac{\Delta_{T,1}}{(c-1)\delta} - 1\right).$$

We now present a first efficiently computable bound on the number of samples s required to have that C'_M is a relative ε-approximation of C_M with probability $\ge 1 - \eta$. Such bound is based on the application of Hoeffding's inequality (see [19]) an advanced but commonly used technique in the analysis of probabilistic algorithms.

Theorem 3.1. *Given $\varepsilon \in \mathbb{R}^+, \eta \in (0,1)$ let $X_1, \dots, X_s$ be the random variables associated with the counts computed at iterations $1, \dots, s$, respectively, of PRESTO-A. If $s \ge \frac{\Delta^2_{T,1}}{2(c-1)^2\delta^2\varepsilon^2}\ln\left(\frac{2}{\eta}\right)$, then it holds*

$$\mathbb{P}\left(\left|\frac{1}{s}\sum_{i=1}^{s} X_i - C_M\right| \ge \varepsilon C_M\right) \le \eta$$

that is, PRESTO-A is an (ε, η)-approximation algorithm.

We now show that by using Bennett's inequality (see [33]), a more advanced result on the concentration of the sum of independent random variables, we can derive a bound that is much tighter than the above, while still being efficiently computable. One of the difficulties in applying such result to our scenario is that we know only an upper bound to the variance of the random variables $X_i, i = 1, \dots, s$. A discussion of why Bennett's inequality can be applied in such situation is in [34]. We now state our main result.

Theorem 3.2. *Given $\varepsilon \in \mathbb{R}^+, \eta \in (0,1)$ let $X_1, \dots, X_s$ be the random variables associated with the counts computed at iterations $1, \dots, s$, respectively, of PRESTO-A. If $s \ge \left(\frac{\Delta_{T,1}}{(c-1)\delta} - 1\right)\frac{1}{(1+\varepsilon)\ln(1+\varepsilon) - \varepsilon}\ln\left(\frac{2}{\eta}\right)$, then it holds*

$$\mathbb{P}\left(\left|\frac{1}{s}\sum_{i=1}^{s} X_i - C_M\right| \ge \varepsilon C_M\right) \le \eta$$

that is, PRESTO-A is an (ε, η)-approximation algorithm.

Note that the bound in Theo. 3.2 is significantly better than the one in Theo. 3.1, since the former has a quadratic dependence from $\Delta_{T,1}$ while the latter enjoys a linear dependence from $\Delta_{T,1}$. Furthermore, differently from the bounds in [29], our bounds depend on characteristic quantities of the datasets ($\Delta_{T,1}$) and of the algorithm's input (c, δ). Therefore we expect our bounds to be more informative, and also possibly tighter than the bounds of [29] (with the improvement due, at least in part, to the use of Bennett's inequality, while [29] leverages on Chebyshev's inequality, which usually provides looser bounds [19]).

3.3 PRESTO-E: Sampling the Start from Edges In this section we present and analyse our second (ε, η) approximation algorithm, PRESTO-E, obtained with a different variant of the general strategy of Algorithm 1.

PRESTO-E selects the starting point t_r (Line 2 in Alg. 1) of sample T_i only from the timestamps of edges of T. In particular, t_r is chosen uniformly at random in $\{t_1, \dots, t_{last}\} \subseteq \{t_1, \dots, t_m\}$, where $t_{last} = \min\{t : (x, y, t) \in E \wedge t \ge t_m - c\delta\}$. When the edges of T are far from each other or have a skewed distribution in time, or occur mostly in some well-spaced subsets $\{[t_a, t_b] \subseteq [t_1, t_m] : t_a \le t_b\}$, we expect PRESTO-E to collect samples with more δ-instances than PRESTO-A (which samples t_r uniformly at random almost from the entire time interval $[t_1, t_m]$ but without restricting to existing edges). Similarly to PRESTO-A, the weight $w(u)$ (Line 8 of Alg. 1) is computed by $w(u) = 1/\tilde{p}_u$, where $\tilde{p}_u$ is the probability that $u \in \mathcal{U}$ is in the set $\mathcal{S}_i$. Due to the (random) choice of t_r, $\tilde{p}_u = \tilde{r}_u/\Delta_{T,2}$, where $\Delta_{T,2} = |\{t : (x, y, t) \in E \wedge t \in [t_1, t_{last}]\}|$ is the number of possible choices for t_r (if $t_{last} = t_m$ then $\Delta_{T,2} = m$) and $\tilde{r}_u = |\{t : (x, y, t) \in E \wedge t \in [\max\{t_1, t_\ell^u - c\delta\}, \min\{t_{last}, t_1^u\}]\}|$ is the number of choices for t_r such that $u \in \mathcal{U}$ is in $\mathcal{S}_i, i = 1, \ldots, s$.

Similarly to what done for PRESTO-A we prove that PRESTO-E provides an unbiased estimate of C_M, with bounded variance (the proof is in [34]). By applying Bennett's inequality, we derive the following.

THEOREM 3.3. *Given $\varepsilon \in \mathbb{R}^+, \eta \in (0, 1)$ let $X_1, \ldots, X_s$ be the random variables associated with the counts computed at iterations $1, \ldots, s$, respectively, of PRESTO-E. If $s \geq \frac{(\Delta_{T,2}-1)}{(1+\varepsilon)\ln(1+\varepsilon)-\varepsilon} \ln\left(\frac{2}{\eta}\right)$, then it holds*

$$\mathbb{P}\left(\left|\frac{1}{s}\sum_{i=1}^{s} X_i - C_M\right| \geq \varepsilon C_M\right) \leq \eta$$

that is, PRESTO-E is an (ε, η)-approximation algorithm.

As for PRESTO-A, by using Bennett's inequality we obtain a linear dependence between the sample size s and $\Delta_{T,2}$, while commonly used techniques (e.g., Hoeffding's inequality) lead to a quadratic dependence.

3.4 PRESTO: Complexity Analysis In this section we analyse the time complexity of PRESTO's versions introduced in the previous sections.

Note that our algorithms employ an exact enumerator as subroutine. For the sake of the analysis we consider the complexity when the subroutine is the algorithm by Mackey et al. [15], which we used in our implementation. Let us first start with a definition, given a temporal network $T = (V, E)$, we denote with $\dim(T)$ the set $\{t_i | (x, y, t_i) \in E, i = 1, \ldots, m\}$. The algorithm in [15] has a worst-case complexity $O(m\hat{\kappa}^{(\ell-1)})$ when executed on a motif with ℓ edges and a temporal network with m temporal edges, where $\hat{\kappa}$ is the maximum number of edges within a window of length δ, i.e., $\hat{\kappa} = \max\{|S(t)| : S(t) = \{(x, y, \bar{t}) \in E : \bar{t} \in [t, t+\delta]\}, t \in \dim(T)\}$. Note that such complexity is *exponential* in the number of edges ℓ of the temporal motif. Obviously our algorithms benefit from any improvement to the state-of-the-art exact algorithms.

Complexity of PRESTO-A. The worst-case complexity is $O(s\hat{m}\hat{\kappa}^{(\ell-1)} + sC_M^*)$ when executed sequentially, where $\hat{m}$ is the maximum number of edges in a window of length $c\delta$, i.e., $\hat{m} = \max\{|S(t)| : S(t) = \{(x, y, \bar{t}) \in E : \bar{t} \in [t, t + c\delta]\}, t \in \dim(T)\}$, and C_M^* is the maximum number of δ-instances contained in any window of length $c\delta$. This corresponds to the case where PRESTO-A collects samples with many edges for which many δ-instances occur. The complexity becomes $O(\frac{s}{\tau}\hat{m}\hat{\kappa}^{(\ell-1)} + \frac{s}{\tau}C_M^*)$ when PRESTO-A is executed in a parallel environment with τ threads (parallelizing the `for` cycle in Alg. 1).

Complexity of PRESTO-E. Using the notation defined above, the worst-case complexity of a naive implementation of PRESTO-E is $O(s\hat{m}\hat{\kappa}^{(\ell-1)} + s\hat{m}C_M^*)$. As for PRESTO-A, the first term comes from the complexity of the exact algorithm for computing $\mathcal{S}_i, i = 1, \ldots, s$, whereas the second term is the worst-case complexity of computing the weights for each sample. The additional $O(\hat{m})$ complexity of the second term arises from the computation of $\tilde{r}_u$ for each $u \in \mathcal{S}_i, i = 1, \ldots, s$. The complexity of this computation can be reduced to $O(\log(m))$ by applying binary search to the edges of T. With such an approach, we obtained the final complexity $O(s\hat{m}\hat{\kappa}^{(\ell-1)} + s\log(m)C_M^*)$. The complexity reduces to $O(\frac{s}{\tau}\hat{m}\hat{\kappa}^{(\ell-1)} + \frac{s}{\tau}\log(m)C_M^*)$ when τ threads are available for a parallel execution.

4 Experimental evaluation

In this section we present the results of our extensive experimental evaluation on large scale datasets. To the best of our knowledge we consider the largest dataset, in terms of number of temporal edges, ever used for the motif counting problem with more than 2.3 billion temporal edges. After describing the experimental setup and implementation (Sec. 4.1), we first compare the quality of the estimated counts provided by PRESTO with the estimates from state-of-the-art sampling algorithms from Liu et al. [13], and Wang et al. [29] (Sec. 4.2). Then we compare the memory requirements of the algorithms (Sec. 4.3), which may be a limiting factor for the analysis of very large datasets. Additional experiments on PRESTO's running time comparison with the exact algorithm by Mackey et al. [15], with the current state of-the art sampling algorithms, and on PRESTO's parallel implementation are in [34].

4.1 Experimental Setup and Implementation

The characteristics of the datasets we considered are in Table 1. EquinixChicago [1] is a *bipartite* temporal network that we built. A detailed description is in [34]. Descriptions of the other datasets are in [20, 9, 13].

We implemented our algorithms in C++14, using the algorithm by Mackey et al. [15] as subroutine for the exact solution of the counting problem on samples. We ran all experiments on a 64-core Intel Xeon E5-

Table 1: Datasets used in our experimental evaluation. We report: number of nodes n; number of edges m; *precision* of the timestamps; *timespan* of the network.

Name	n	m	Precision	Timespan
Stackoverflow (SO)	2.58M	47.9M	sec	2774 (days)
Bitcoin (BI)	48.1M	113M	sec	2585 (days)
Reddit (RE)	8.40M	636M	sec	3687 (days)
EquinixChicago (EC)	11.16M	2.32B	μ-sec	62.0 (mins)

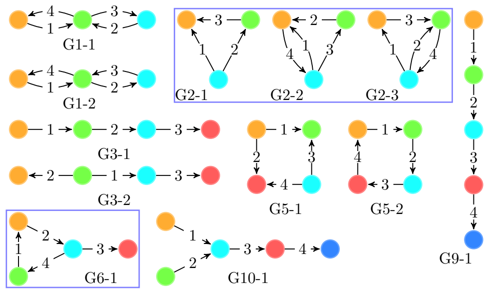


Figure 2: Motifs used in the experimental evaluation. The edge labels denote the edge order in σ according to Definition 2.2.

2698 with 512GB of RAM machine, with Ubuntu 14.04. The code we developed, and the dataset we build are entirely available at `https://github.com/VandinLab/PRESTO/`, links to additional resources (datasets and other implementations) are in [34]. We tested all the algorithms on the motifs from Figure 2, that are a temporal version of graphlets from [23], often used in the analysis of static networks. We considered motifs with at most $\ell = 4$ edges since the implementation by Wang et al. [29] does not allow for motifs with a higher ℓ in input. Since the EC dataset is a bipartite network, it can contain all but the motifs marked with a rectangle in Fig. 2. Therefore, for such dataset we did not consider the motifs in the rectangles of Fig. 2. We compared our algorithms `PRESTO-A` and `PRESTO-E`, collectively denoted as `PRESTO`, with 2 baselines: the sampling algorithm by Liu et al. [13], denoted by `LS` and the sampling algorithm by Wang et al. [29], denoted by `ES`.

4.2 Quality of Approximation We start by comparing the approximation provided by `PRESTO` with the current state-of-the-art sampling algorithms. To allow for a fair comparison, since the algorithms we consider depend on parameters that are not directly comparable and that influence both the algorithm's running time and approximation, we run all the algorithms sequentially and measure the approximation they achieve by fixing their running time (i.e., we fix the parameters to obtain the same running time).

While `PRESTO` can be trivially modified in order to work within a user-specified time limit, the same does not apply to `LS` or `ES`. Thus, to fix the same running time for all algorithms we devised the following procedure: i) for a fixed dataset and δ, we found the parameters of `LS` such its runtime is at least one order of magnitude[2] smaller than the exact algorithm by Mackey et al. [15] on motif G1-1 ii) we run `LS` on all the motifs on the fixed dataset with the parameters set as from i) and measure the time required for each motif; iii) we run `ES` such that it requires the same time[3] as `LS`; iv) we ran `PRESTO-A` and `PRESTO-E` with $c = 1.25$ and a time limit that is the *minimum* between the running time of `LS` and the one of `ES`. The parameters for all algorithms are in [34]. Further discussion on how to set c when running `PRESTO` is in [34].

For each configuration, we ran each algorithm ten times, with a limit of 400 GB of (RAM) memory. We computed the so called Mean Average Percentage Error (MAPE) after removing the best and worst results, to remove potential outliers. MAPE is defined as follows: let C'_M be the estimate of the count C_M produced by one run of an algorithm, then the relative error of such estimate is $|C'_M - C_M|/C_M$; the MAPE of the estimates is the average of the relative errors in percentage.

Table 2 shows the MAPE of the sampling algorithms on SO, BI, and RC datasets (due to space constraints we report the variances in [34]). On the SO dataset we observe that both versions of `PRESTO` provide more accurate estimates over current state-of-the-art sampling algorithms. `PRESTO-A` provides the best approximation on 6 out of 12 motifs, with `PRESTO-E` providing the best results in all the other cases except for motif G2-1. Furthermore, both variants of `PRESTO` improve on all motifs w.r.t. `LS` and on 11 out of 12 motifs w.r.t. `ES`, with the error from `ES` being more than three times the error of `PRESTO` on such motifs. We note `PRESTO-A` and `PRESTO-E` achieve similar results, supporting the idea that similar performances are obtained when the network's timestamps are distributed evenly, that is the case for such dataset (see [34]).

For the BI dataset the results are similar to the ones for the SO dataset. `PRESTO-A` achieves the lowest approximation error on 7 out of 12 motifs with `PRESTO-E` achieving the lowest approximation error on

[2]We use such threshold since sampling algorithms are often required to run within a small fraction of time w.r.t. exact algorithms.

[3]`ES` needs to preprocess each dataset before executing the sampling procedure. To account for this preprocessing step, we measured the time for preprocessing as geometric average across ten runs, and added to `ES`'s running time such value divided by the number of motifs (i.e., 12).

Table 2: Comparison of the relative approximation error. For each combination of dataset and motif we highlight the algorithm with minimum MAPE value.

	SO					BI					RE				
		Approximation Error					Approximation Error					Approximation Error			
		PRESTO					PRESTO					PRESTO			
Motif	C_M	A	E	LS	ES	C_M	A	E	LS	ES	C_M	A	E	LS	ES
G10-1	$7.8 \cdot 10^8$	**4.1%**	4.2%	4.2%	12.9%	$1.2 \cdot 10^{10}$	14.2%	**13.1%**	25.8%	16.9%	$5.8 \cdot 10^9$	14.5%	**9.6%**	14.1%	32.8%
G1-1	$2.1 \cdot 10^6$	3.3%	**1.9%**	7.8%	23.3%	$6.8 \cdot 10^7$	**7.1%**	15.8%	21.2%	44.9%	$1.5 \cdot 10^8$	25.3%	25.2%	30.7%	**17.1%**
G1-2	$8.3 \cdot 10^5$	6.6%	**2.9%**	8.7%	62.0%	$5.7 \cdot 10^7$	**9.3%**	15.5%	25.1%	59.7%	$1.2 \cdot 10^8$	43.8%	**24.6%**	31.3%	65.3%
G2-1	$7.7 \cdot 10^5$	4.4%	3.9%	8.6%	**1.6%**	$2.3 \cdot 10^6$	**5.0%**	7.1%	21.4%	30.5%	$3.3 \cdot 10^7$	7.4%	**4.3%**	8.4%	20.1%
G2-2	$3.5 \cdot 10^5$	**3.2%**	4.6%	9.7%	53.4%	$4.9 \cdot 10^5$	11.0%	26.8%	34.6%	**3.4%**	$2.9 \cdot 10^7$	39.4%	46.1%	33.4%	**9.8%**
G2-3	$7.0 \cdot 10^5$	**9.3%**	9.4%	10.1%	60.7%	$9.9 \cdot 10^5$	**7.9%**	22.7%	19.6%	46.9%	$9.4 \cdot 10^7$	18.4%	16.9%	**16.7%**	32.6%
G3-1	$2.3 \cdot 10^8$	**1.6%**	2.7%	4.6%	9.8%	$7.3 \cdot 10^8$	2.5%	**2.2%**	21.9%	13.7%	$4.2 \cdot 10^8$	3.1%	**1.8%**	3.3%	10.2%
G3-2	$2.3 \cdot 10^8$	2.9%	**1.8%**	5.3%	11.3%	$5.7 \cdot 10^8$	3.6%	**1.7%**	21.8%	17.6%	$5.3 \cdot 10^8$	2.7%	**1.2%**	3.1%	7.5%
G5-1	$8.0 \cdot 10^5$	**4.0%**	4.2%	5.0%	22.3%	$5.1 \cdot 10^6$	**14.5%**	17.6%	16.9%	32.4%	$8.2 \cdot 10^7$	8.1%	**7.0%**	8.2%	31.9%
G5-2	$5.0 \cdot 10^5$	**4.2%**	5.0%	5.4%	34.4%	$1.3 \cdot 10^6$	23.2%	**17.6%**	23.3%	33.8%	$1.7 \cdot 10^7$	28.1%	**13.8%**	25.9%	52.0%
G6-1	$2.0 \cdot 10^6$	6.4%	**6.1%**	12.7%	47.6%	$1.0 \cdot 10^7$	**7.0%**	16.7%	19.4%	37.3%	$5.5 \cdot 10^7$	33.2%	30.7%	**20.0%**	50.0%
G9-1	$4.1 \cdot 10^8$	4.4%	**2.5%**	6.0%	10.5%	$5.8 \cdot 10^8$	**2.3%**	3.3%	26.7%	11.8%	$6.0 \cdot 10^8$	**3.8%**	5.4%	4.5%	19.9%

all the other motifs but G2-2. `PRESTO-A` achieves better estimates than `LS` on all motifs and on 11 out of 12 motifs w.r.t. `ES`, `PRESTO-E` behaves similarly, except that it improves over `LS` on 10 out of 12 motifs. On the RE dataset, `PRESTO-E` achieves the lowest approximation error on 7 out of 12 motifs and it improves over `PRESTO-A` for most of the motifs. This is not surprising, since [29] showed that RE has a more skewed edge distribution (which we also discuss in [34]), a scenario for which we expect `PRESTO-E` to improve over `PRESTO-A` (see Sec. 3.3). Nonetheless, `PRESTO-A` achieves lower approximation error over 6 motifs when comparing to `LS` and on 10 motifs w.r.t. `ES`, while `PRESTO-E` achieves the best approximation error on most of the motifs, improving over `LS` on 8 motifs and over `ES` on 10 motifs.

Finally, Table 3 shows the results on the EC dataset, which is a 2.3 billion edges *bipartite* temporal network. Note that the results for `ES` are missing, since `ES` did not complete any run with 400GB of memory on the motifs tested. (We discuss the high memory usage of `ES` in Sec. 4.3). On such dataset both variants of `PRESTO` perform better than `LS` on 7 out 8 motifs and achieve the lowest approximation error on 4 out 8 motifs each. Interestingly, the variance of `PRESTO` is often similar or lower than the one of `LS`, which is another of the advantages of our algorithms. We observed similar trends, and even smaller variances for `PRESTO` across all the experiments on the other datasets (see [34]).

These results, coupled with the theoretical guarantees of Sec. 3, show that `PRESTO` outperforms the state-of-the-art sampling algorithms `LS` and `ES` for the estimation of most of the motif counts on temporal networks. Based on the results we discussed, `PRESTO-E` seems also to usually provide similar estimates to `PRESTO-A`, while significantly improving over `PRESTO-A` when the network edges are not uniformly distributed such the `EC` dataset.

Table 3: Approximation error results on EC dataset.

Motif	C_M	PRESTO-A	PRESTO-E	LS
G10-1	$6.3 \cdot 10^{10}$	**2.8±2.8%**	4.2±3.2%	4.7±1.9%
G1-1	$1.5 \cdot 10^{11}$	7.6±4.1%	**5.1±3.4%**	8.0±3.7%
G1-2	$1.5 \cdot 10^{11}$	6.5±4.1%	**4.6±2.4%**	7.8±3.0%
G3-1	$1.6 \cdot 10^{10}$	**1.9±1.5%**	3.5±1.8%	3.0±1.8%
G3-2	$1.6 \cdot 10^{10}$	**2.6±1.0%**	2.8±2.0%	3.7±2.2%
G5-1	$3.4 \cdot 10^{11}$	10.4±7.0%	**8.2±4.6%**	14.9±5.4%
G5-2	$3.8 \cdot 10^{10}$	14.2±5.4%	**4.5±4.0%**	9.0±5.0%
G9-1	$8.7 \cdot 10^{10}$	**6.3±3.8%**	6.8±2.2%	9.9±4.6%

Table 4: Minimum and peak memory usage in GB over all motifs in Figure 2, "✗" denotes out of memory.

Dataset	PRESTO-A	PRESTO-E	LS	ES
SO	1.5 - 1.5	1.5 - 1.5	1.5 - 1.5	9.7 - 10.5
BI	3.5 - 3.5	3.5 - 3.7	3.5 - 4.2	28.8 - 29.5
RE	19.5 - 19.8	19.5 - 19.8	19.5 - 20.8	129.9 - 141.6
EC	71.0 - 71.0	71.0 - 71.0	71.0 - 71.0	✗

4.3 Memory usage In this section we discuss the RAM usage of the various sampling algorithms. The results are in Table 4, which shows for each algorithm the minimum and maximum amount of (RAM) memory used over all motifs in Figure 2 (collected on a single run). Both `PRESTO`'s versions have lower memory requirements than `ES`, and equal or lower memory requirements than `LS`, on all datasets. `ES` ranges from requiring 6.5× up to 8.5× the memory used by `PRESTO`, making `ES` not practical for very large datasets such as EC (where `ES` did not completed any of the runs since it exceeded the 400GB of memory allowed for its execution). While `LS` and `PRESTO` have similar memory requirements, `LS` displays greater variance on BI and RC, which may be due to the fact that choosing windows with more edges, as done by `LS`, may require more memory to run the exact algorithm as subroutine. `PRESTO` is, thus, a memory efficient algorithm and hence a very practical tool to tackle the motif counting problem on temporal networks with billions edges.

5 Conclusions

In this work we introduced `PRESTO`, a simple yet practical algorithm for the rigorous approximation of temporal motif counts. Our extensive experimental evaluation shows that `PRESTO` provides more accurate results and is much more scalable than the state-of-the-art sampling approaches. There are several interesting directions for future research, including developing rigorous techniques for mining several motifs simultaneously and developing efficient algorithms to maintain estimates of motif counts in adversarial streaming scenarios.

References

[1] The CAIDA UCSD Anonymized Internet Traces – February 17, 2011. `https://www.caida.org/data/passive/passive_2011_dataset.xml`.

[2] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella. Graph sample and hold: A framework for big-graph analytics. *KDD* 2014.

[3] F. Battiston, V. Nicosia, M. Chavez, and V. Latora. Multilayer motif analysis of brain networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 2017.

[4] A. R. Benson, D. F. Gleich, and J. Leskovec. Higher-order organization of complex networks. *Science*, 2016.

[5] M. Bressan, F. Chierichetti, R. Kumar, S. Leucci, and A. Panconesi. Counting graphlets: Space vs time. *WSDM* 2017.

[6] M. Ceccarello, C. Fantozzi, A. Pietracaprina, G. Pucci, and F. Vandin. Clustering uncertain graphs. *PVLDB* 2017.

[7] L. De Stefani, E. Terolli, and E. Upfal. Tiered sampling: An efficient method for approximate counting sparse motifs in massive graph streams. *Big Data* 2017.

[8] J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques.* Elsevier, 2011.

[9] J. Hessel, C. Tan, and L. Lee. Science, AskScience, and BadScience: On the Coexistence of Highly Related Communities. *AAAI ICWSM* 2016.

[10] P. Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 2015.

[11] P. Holme and J. Saramäki. Temporal networks. *Physics reports*, 2012.

[12] P. Holme and J. Saramäki, editors. *Temporal Network Theory.* Comp. Social Sciences. Springer, 2019.

[13] P. Liu, A. R. Benson, and M. Charikar. Sampling Methods for Counting Temporal Motifs. *WSDM* 2019.

[14] P. Liu, V. Guarrasi, and A. E. Sarıyüce. Temporal network motifs: Models, limitations, evaluation. *arXiv:2005.11817*, 2020.

[15] P. Mackey, K. Porterfield, E. Fitzhenry, S. Choudhury, and G. Chin. A Chronological Edge-Driven Approach to Temporal Subgraph Isomorphism. *Big Data* 2018.

[16] S. Mangan and U. Alon. Structure and function of the feed-forward loop network motif. *PNAS*, 2003.

[17] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon. Superfamilies of evolved and designed networks. *Science*, 2004.

[18] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 2002.

[19] M. Mitzenmacher and E. Upfal. *Probability and computing.* Cambridge University Press, 2017.

[20] A. Paranjape, A. R. Benson, and J. Leskovec. Motifs in Temporal Networks. *WSDM* 2017.

[21] H.-M. Park, F. Silvestri, U. Kang, and R. Pagh. Mapreduce triangle enumeration with guarantees. *CIKM* 2014.

[22] T. P. Peixoto and L. Gauvin. Change points, memory and epidemic spreading in temporal networks. *Sci. rep.*, 2018.

[23] N. Pržulj, D. G. Corneil, and I. Jurisica. Modeling interactome: scale-free or geometric? *J. Bioinf.*, 2004.

[24] L. D. Stefani, A. Epasto, M. Riondato, and E. Upfal. Triest: Counting local and global triangles in fully dynamic streams with fixed memory size. *TKDD*, 2017.

[25] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. *KDD* 2007.

[26] C. E. Tsourakakis, U. Kang, G. L. Miller, and C. Faloutsos. Doulion: counting triangles in massive graphs with a coin. *KDD* 2009.

[27] K. Tu, J. Li, D. Towsley, D. Braines, and L. D. Turner. Network classification in temporal networks using motifs. *arXiv:1807.03733*, 2018.

[28] J. Ugander, L. Backstrom, and J. Kleinberg. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. *WWW* 2013.

[29] J. Wang, Y. Wang, W. Jiang, Y. Li, and K.-L. Tan. Efficient sampling algorithms for approximate temporal motif counting. *arXiv:2007.14028*, 2020.

[30] J. Wu, J. Liu, W. Chen, H. Huang, Z. Zheng, and Y. Zhang. Detecting mixing services via mining bitcoin transaction network with hybrid motifs. *arXiv:2001.05233*, 2020.

[31] Ö. N. Yaveroğlu, N. Malod-Dognin, D. Davis, Z. Levnajic, V. Janjic, R. Karapandza, A. Stojmirovic, and N. Pržulj. Revealing the hidden language of complex networks. *Sci. rep.*, 2014.

[32] C. Belth, X. Zheng and D. Koutra. Mining Persistent Activity in Continually Evolving Networks. *KDD*, 2020.

[33] S. Boucheron, G. Lugosi and P. Massart. *Concentration inequalities: A nonasymptotic theory of independence.* Oxford university press, 2013.

[34] I. Sarpe and F. Vandin. PRESTO: Simple and Scalable Sampling Techniques for the Rigorous Approximation of Temporal Motif Counts. *arXiv:2101.07152*, 2021.