symmetry

MDPI

Article

New Algorithms for Counting Temporal Graph Pattern

Xiaoli Sun [1,*], Yusong Tan [1], Qingbo Wu [1], Jing Wang [1] and Changxiang Shen [2]

[1] College of computer, National University of Defence Technology, Changsha 410073, China; tanyusong@kylinos.cn (Y.T.); wuqingbo@kylinos.cn (Q.W.); wangjing@kylinos.cn (J.W.)
[2] College of computer, Beijing University of Technology, Beijing 100000, China; shenchx@cae.cn
* Correspondence: sunxiaoli12@nudt.edu.cn

Received: 25 August 2019; Accepted: 17 September 2019; Published: 20 September 2019

Abstract: Temporal networks can describe multiple types of complex systems with temporal information in the real world. As an effective method for analyzing such network, temporal graph pattern (TGP) counting has received extensive attention and has been applied in diverse domains. In this paper, we study the problem of counting the TGP in the temporal network. Then, an exact algorithm is proposed based on the time first search (TFS) algorithm. This algorithm can reduce the intermediate results generated in the graph isomorphism and has high computational efficiency. To further improve the algorithm performance, we design an estimation algorithm by applying the edge sampling strategy to the exact algorithm. Finally, we evaluate the performances of the two algorithms by counting both the symmetric and asymmetric TGP. Extensive experiments on real datasets demonstrated that the exact algorithm is faster than the existing algorithm and the estimation algorithm can greatly reduce the running time while guaranteeing the accuracy.

Keywords: temporal network; temporal graph pattern (TGP); TGP counting; edge sampling; TFS

1. Introduction

A graph is a tool for describing the complex systems, such as biological [1], chemical [2], and electronic interactive systems [3]. In many cases, these systems are accompanied by the temporal information. One example is that in the social network the time when people send the messages is always recorded [4]. Another example is that the chronological order of the packet transmission is also concerned in cyber network [5]. Obviously, the temporal information is a key component of such systems. For the purpose of analysis, the complex systems are usually represented by the networks (graphs). However, the static networks cannot describe the complex systems that contain the temporal information. Therefore, the temporal network, in which each edge has a timestamp, is proposed [6].

Graph pattern counting is a fundamental problem in network analysis, including anomaly detection [7], community detection [8], internet traffic classification [9] and so on. Usually, the graph pattern refers to small and induced subgraph, which is also called motif or graphlet. In particular, for the temporal network, if the graph pattern has temporal information, then the pattern becomes temporal graph pattern (TGP). TGP has been widely studied and is more commonly known as temporal motif in many studies. According to the application scenario, there are two kinds of definitions about temporal motif. The first definition is the network motif for the graph snapshots at different time points, in which the edges have continuous timestamps [10]. The second definition is the temporal motif of a whole temporal graph. In this paper, we consider the second type of temporal motif.

The counting problem of small graph pattern in static network has been proposed for a long time and attracted widespread attention, especially the triangle counting in social network analysis [11–13]. Until now, most of the triangle counting algorithms are estimation algorithms, and the graph stream

model has been applied in the algorithms to optimize memory usage [12,14]. In recent years, more and more scholars have shown interest in the problem of counting a large graph pattern (i.e., the pattern with more than three vertexes) [15–17]. The counting algorithms can be divided into two categories: the exact algorithms and the estimation algorithms. The former is designed for the graph pattern with no more than five vertexes and has low efficiency [15–21]. Even for the fastest exact algorithm, the Efficient Subgraph Counting Algorithmic PackagE (ESCAPE) [15], it takes a week to count the five-vertex graph pattern in a graph with a million vertexes. Compared to the exact algorithms, the estimation algorithms reduce the search space by sampling the graph and thus improve the computational efficiency [18–21]. Rahman et al. [22] reduced the counting time by sampling the edges. Bhuiyan et al. [21] used a Markov chain to uniformly sample motifs and obtained the counts of three-, four-, and five-vertex graphlets. In the existing estimation algorithm, the most frequently used sampling methods are path sampling [23], random walk [19] and color coding [20,24], where the path sampling can not be applied to count the graph pattern with more than five vertexes, while the other two methods can be used for larger graph patterns.

The above algorithms are designed for graph pattern counting in static network. They cannot be directly applied to the temporal network because the problems in temporal network and static network are totally different. Firstly, the temporal network has temporal information, while the static network does not. Secondly, the edges of the TGP have a chronological order, while the edges of the static graph pattern do not. Thirdly, the graph isomorphism of the temporal network, which is more complex than that of the static network, considering not only the topology structure isomorphism but also the chronological order of the edges in the temporal graph.

At present, there are many studies on the graph pattern counting problem in temporal network. At the early stage, TGP (known as temporal motif) was usually defined as the network motif of graph snapshots at different time points [10,25]. On the basis of this definition, Bajardi et al. [25] further defined the dynamic motif of the cattle trade movement. Recently, temporal motif is no longer limited by snapshots, but extends to the network motifs with temporal attribute [26]. Inspired by this idea, Kovanan et al. [27] proposed a temporal motif definition that is widely used in Wikipedia network [28] and combat system coordination [29], and Paranjape et al. [30] put forward the δ-temporal motif. Although both definitions have temporal attribute, there are still some differences between them. On the one hand, δ-temporal motif stipulates that the edges must be sorted according to the timestamps, i.e., there is no edge of the same time, while the definition in [27] may have several edges with the same time. On the other hand, all the edges of δ-temporal motif should be in a fixed time window, while the temporal motif in [27] only requires that the time difference between adjacent edges does not exceed the threshold. Afterwards, Mackey et al. [31] used the δ-temporal motif and designed a chronological edge-driven method to search all matched subgraphs of a given temporal graph. Liu et al. [32] proposed a sampling framework for counting the δ-temporal motif. Since these counting methods are based on the δ-temporal motif, it is hard for them to count the temporal motif in [27].

Although there are many algorithms for counting different kinds of TGP (or temporal motifs), the TGP counting still has two challenges. Firstly, the counting algorithm for the TGP defined in [27] is lacking. Secondly, the existing algorithms do not consider the TGP that has multiple edges with the same time. To solve the challenges, here we study the counting problem for the TGP defined in [27], and propose an exact algorithm and an estimation algorithm. The exact algorithm first partitions the graph into several temporal subgraphs according to the time threshold, and then counts the TGP of each temporal subgraph. The key problem in the counting process is the temporal graph isomorphism, which is actually a typical NP-complete problem. To fix this problem, we produce an edge order via the time first search (TFS) algorithm, and match the temporal motif according to the order. Since the TFS algorithm considers both the temporal and topological information simultaneously, the intermediate results in the isomorphism are reduced and the efficiency of the algorithm is improved. Based on the exact algorithm, the estimation algorithm which can achieve an unbiased estimation is proposed. Because of the use of edge sampling, this algorithm can greatly reduce the running time

while guaranteeing the accuracy. Moreover, both algorithms are suitable for the TGP that has multiple edges with the same time. The main contributions of this paper are as follows:

(1) The paper studies the problem of counting the temporal graph pattern defined in [27] and proposes the corresponding problem model. Since all the counting algorithms in the existing literature are designed for the TGP defined in [30], the problem here has not been discussed before and is of great significance.

(2) The paper provides a strategy of the TFS algorithm to process the temporal graph pattern that has multiple edges with the same time. The TFS algorithm gives an edge order determined according to the topology when there exist such edges. The edge order is used to match the edges, and makes the proposed algorithms suitable for any TGP.

(3) The paper proposes an exact algorithm and an estimation algorithm to count the TGP in temporal graph. Both algorithms are computationally efficient because they can match the topology and temporal information simultaneously.

The rest of this manuscript is organized as follows. Section 2 gives some definitions and describes the temporal graph counting problem for the temporal network. In Section 3, we propose an exact algorithm and an estimation algorithm for the TGP counting. Section 4 presents a variety of experiments to demonstrate the effectiveness of the algorithms. Section 5 gives some discussions. Finally, conclusions are provided in Section 6.

2. Definitions and Problem

In this section, we give the fundamental definitions of the temporal graph, the temporal graph pattern, the temporal graph isomorphism and so on. Then, according to these definition, we briefly describe the counting problem of the temporal graph pattern.

Definition 1. ***Temporal Graph.*** *A temporal graph $G = (V, E)$ consists of a set of vertices V and a set of temporal edges $E = \{(u, v, t) | u, v \in V\}$, where t is the timestamp of the edge.*

Definition 2. ***ΔT-Temporally Related Edges.*** *Given two edges $\boldsymbol{e_i} = (\boldsymbol{u_i}, \boldsymbol{v_i}, \boldsymbol{t_i})$ and $\boldsymbol{e_j} = (\boldsymbol{u_j}, \boldsymbol{v_j}, \boldsymbol{t_j})$, the edge $\boldsymbol{e_i}$ is $\boldsymbol{\Delta T}$-temporally related to edge $\boldsymbol{e_j}$ if they are temporally adjacent, i.e., $\{\boldsymbol{u_i}, \boldsymbol{v_i}\} \cap \{\boldsymbol{u_j}, \boldsymbol{v_j}\} \neq \emptyset$ and $|\boldsymbol{t_i} - \boldsymbol{t_j}| \leq \boldsymbol{\Delta T}$.*

Definition 3. ***ΔT-Temporally Connected Graph.*** *A temporal graph $G = (V, E)$ is ΔT-temporally connected graph if and only if the graph is weakly connected and all the adjacent edges are ΔT-temporally related edges.*

Definition 4. ***Temporal Graph Pattern.*** *A temporal graph pattern $H = (g, \Delta T)$, also known as the temporal motif, is a ΔT-temporally connected graph $g = (V_h, E_h)$.*

Definition 4 is just one of the TGP definitions, and we aim at counting the number of this pattern in a large temporal graph. Since the TGP contains the temporal information, the graph isomorphism needs to satisfy both the isomorphic conditions of static graph and the temporal conditions. In the following, we further give the definition of temporal graph isomorphism and the conditions that need to be satisfied.

Definition 5. ***Temporal Graph Isomorphism.*** *If a temporal subgraph $G = (V, E)$ is temporally isomorphic to the temporal graph pattern $H = (g, \Delta T)$, where $g = (V_h, E_h)$, then there exists an injective function*

$\varphi : V_h \to V$ *which satisfies the following conditions:*
(1) $\forall u_h \in V_h, \exists \varphi(u_h) \in V$
(2) $\forall\ e_h = (u_h, v_h, t_h) \in E_h, \exists e = (\varphi(u_h), \varphi(v_h), t) \in E$
(3) **For** $e_{h_i} = (u_{h_i}, v_{h_i}, t_{h_i}), e_{h_j} = (u_{h_j}, v_{h_j}, t_{h_j}) \in E_h$
if $t_{h_i} < t_{h_j}$
then $\exists\ e_i = (\varphi(u_{h_i}), \varphi(v_{h_i}), t_i), e_j = (\varphi(u_{h_j}), \varphi(v_{h_j}), t_j) \in E$
s.t. $t_i < t_j$.

Condition (3) describes the temporal relationship among the edges. In addition to the above conditions, all the adjacent edges in the matched subgraph G must be the ΔT-temporally related edges.

Therefore, given a temporal graph G and the TGP H, the problem here is to count the number of subgraphs that are temporally isomorphic to H in G.

3. The Counting Algorithms for TGP

To solve the TGP counting problem, an edge-centric exact algorithm based on the TFS method is first proposed in this section. Then, we combine the exact algorithm with the edge sampling method to reduce the iteration number of the edges, and present an estimation algorithm.

3.1. The Exact Algorithm

3.1.1. The Overall Idea of the Algorithm

As mentioned in the previous section, the temporal motif is a ΔT-temporally connected graph and the time differences among the adjacent edges of the temporal motif should not exceed the time threshold ΔT. Therefore, we can partition the temporal graph into multiple subgraphs according to ΔT, and count the number of the graph pattern in each subgraph. After accumulating the numbers from different subgraphs, the total number of TGP can be obtained. Obviously, the counting problem can be handled by three steps: (1) partitioning the temporal graph; (2) counting the number of TGP in each subgraph; and (3) accumulating the numbers.

In Step (1), we need to traverse all the edges in the temporal graph and ensure that the adjacent edges whose time differences do not exceed the threshold ΔT are placed in the same subgraph. After the partition, we can get a set of temporal subgraphs $S = \{G_{t_i}, i = 0, 1..., M\}$, and each edge in the temporal graph G only exists in one subgraph. The partitioning process can be easily achieved by the algorithm in work [33], thus here we do not describe the partitioning algorithm in detail.

The key step for solving the counting problem is Step (2), i.e., counting the number of TGP in the obtained temporal subgraphs. In this step, the NP-Complete graph isomorphism problem is inevitable. Compared to the isomorphism problem in static graph, the temporal isomorphism problem requires extra consideration of the temporal relationship, which means that the counting problem of TGP in temporal graph is more difficult than the counting problem of the non-temporal graph pattern in static graph. In the following, we focus on Step (2) and then design a counting algorithm that can match the temporal relationship and the topology simultaneously.

3.1.2. Counting the Number of TGP in Each Subgraph

To count the number of TGP, we need to search the isomorphic graphs in the subgraphs G_{t_i}, $i = 0, 1..., M$. A common strategy used in the graph isomorphism is to match the edges exactly according to the chronological order of the edges of TGP. The advantage of this strategy is that it reduces the intermediate results that satisfy the topology but do not satisfy the chronological order. However, since TGP may have several edges with the same time, there may be some intermediate results that satisfy the chronological order but do not satisfy the topology in the process of isomorphism. To eliminate such intermediate results, here we apply the TFS algorithm proposed in our previous work [33] to match the edges.

The TFS algorithm is an edge traversal algorithm, and its output is the traversal order of the edges of the input temporal graph. This algorithm first sorts the edges in the temporal graph by time to select the edge with the smallest time. If there are multiple edges with the smallest time, one of them will be selected randomly. Then, the algorithm traverses the adjacent edges of the selected edge, and selects the new edge with the smallest time. Finally, it repeats the process of selecting the new edge from the adjacent edges of the already selected edges until all the edges of the input temporal graph have been traversed. As the algorithm combines the edge time with the topology, the traversal order satisfies the chronological order of the edges as much as possible while guaranteeing the topological connection.

By applying the above TFS algorithm to the TGP, the traversal order of the TGP can be obtained. Then, we need to get the matched pairs consisting of an edge of temporal graph and a pattern edge one by one according to this order, and count the number of TGP. Algorithm 1 provides the details of the matching process and summarizes the counting algorithm for subgraphs $G_{t_i}, i = 0, 1..., M$. Given the subgraph G_{t_i} obtained from graph partitioning and the TGP H, this algorithm outputs the number of the subgraphs isomorphic to the TGP H in G_{t_i}. First, we sort the edges of H according to the TFS algorithm (Line 1), and get the first edge e_{h_0} of the traversal order L (Line 2). Then, we traverse all the edges of G_{t_i} to get the isomorphic subgraphs (Line 5). If the edge e of the temporal graph G_{t_i} is isomorphic to the edge e_{h_0} (i.e., e satisfies the matching conditions) (Line 6), we add the matched pair P consisting of e and e_{h_0} to the matched pair set M (Lines 7–8), and call the algorithm Match() to match the rest edges of H (Lines 9–11). The algorithm Match() is a recursive algorithm shown in Algorithm 2. In this algorithm, we first determine whether the set M contains all the edges of L (Line 1). If so, the *count* is incremented by 1 and returned (Lines 2–3); otherwise, we continue to find the matched subgraph. Then, we get the next unmatched edge e_{h_j} of L and all the candidate edges E_c from the adjacent edges of matched edges (Lines 5–6). For the edge e' in E_c, if e' satisfies the matching conditions (Lines 7–8), we store the matched pair consisting of e' and e_{h_j} in M and continue to call the match algorithm (Lines 9–12). It is easy to observe that the algorithm Match() returns the number of isomorphic subgraphs whose first matched edge is e. Thus, after traversing all the edges of G_{t_i}, the number of all the isomorphic graphs can be obtained.

Algorithm 1 The counting algorithm for subgraph G_{t_i}.

Input: $G_{t_i} = (V_{t_i}, E_{t_i})$: the temporal subgraph obtained from graph partitioning,
$H = (g, \Delta T)$: the temporal graph pattern.
Output: C_i: the number of the subgraphs isomorphic to H in G_{t_i}.
1: $L \leftarrow$Sort the edges of g according to the TFS algorithm;
2: Get the first edge e_{h_0} of the traversal order L;
3: $M = \emptyset$;
4: $C_i = 0$;
5: **for** edge $e \in G_{t_i}$ **do**
6: **if** e satisfies the matching conditions **then**
7: $P = \text{Pair}(e, e_{h_0})$;
8: $M = M \cup P$;
9: $count = \text{Match}(M, L, G_{t_i}, 0)$;
10: $C_i = C_i + count$;
11: $M = \emptyset$;
12: **end if**
13: **end for**
14: **return** C_i

Algorithm 2 Match($M, L, G, count$).

Input: M: the matched edge pairs, L: the sorted edges of the temporal graph pattern,
G: the temporal graph, $count$: the number of the subgraphs isomorphic to H.
Output: $count$: the number of the subgraphs isomorphic to H.

```
1: if M covers all the edges of the L then
2:     count++;
3:     return count
4: end if
5: e_{h_j} ← Get the next unmatched edge of L;
6: E_c ← Get all the candidate edges from the adjacent edges of matched edges;
7: for edge e' ∈ E_c do
8:     if e' satisfies the matching conditions then
9:         P = Pair(e', e_{h_j});
10:        M = M ∪ P;
11:        Match(M, L, G, count);
12:        Delete(M, P);
13:    end if
14: end for
```

On Line 6 of Algorithm 1 and Line 8 of Algorithm 2, we search the matched edges via the matching conditions. Actually, the matched edges need to satisfy two types of conditions: topological condition and temporal condition. Temporal condition is mainly reflected in two aspects. On the one hand, the time difference between adjacent edges does not exceed the threshold ΔT. On the other hand, the chronological order of the matched edges should be consistent with that of the edges in TGP H. The topological condition restricts the degree of the vertex, which means that the degree of the matched vertex in the temporal graph is equal to or larger than the degree of the vertex in H. In addition, the candidate edges on Line 6 of Algorithm 2 are obtained according to the topological condition. When we search the isomorphic edge of the edge $e_{h_j} = (u_{h_j}, v_{h_j}, t_{h_j}), j > 0$, the source vertex u_{h_j} or the target vertex v_{h_j} should have been matched. Thus, there are three cases in the selection of candidate edges. (1) If only the source vertex u_{h_j} is matched, we traverse the outgoing edges of the matched vertex of the u_{h_j} and choose the edges that satisfy the temporal condition as the candidate edges. (2) If only the target vertex v_{h_j} is matched, we traverse the incoming edges of the matched vertex of the v_{h_j} and choose the candidate edges in the same way. (3) If both the u_{h_j} and the v_{h_j} are matched, we choose the candidate edges from the intersection of the corresponding outgoing edge set and incoming edge set.

3.1.3. Algorithm Summary and Computational Complexity

According to Algorithm 1, we get the number C_i of subgraphs isomorphic to the TGP H in G_{t_i}. Then, by applying this algorithm $M + 1$ times and accumulating all the numbers $C_i, i = 0, ..., M$, the total number C of isomorphic subgraphs can be obtained. Algorithm 3 summarizes the whole counting algorithm (which we call the exact algorithm). The computational complexity of the algorithm is analyzed as follows. First, the computational complexity of the partition processing is $\mathcal{O}(|E|)$, where $|E|$ is the number of the edges in G. Then, the worst complexity of counting H in subgraph G_{t_i} is $\mathcal{O}((m_{t_i})^{m_h})$, where m_{t_i} and m_h are the edge numbers of G_{t_i} and H, respectively. The case would occur when each edge in G_{t_i} could match each edge in H. However, if the edges have a temporal order, the temporal graph isomorphism has exponential asymptotic complexity $\mathcal{O}(m_{t_i}^2)$. Since the whole algorithm contains the partition processing and needs to count the numbers of H in subgraphs $G_{t_i}, i = 0, ..., M$, the total computational complexity of the exact algorithm is $\mathcal{O}(\sum_{i=0}^{M}(m_{t_i})^{m_h} + |E|)$. It is

worth noting that, in real scenarios, most of subgraphs $G_{t_i}, i = 0, ..., M$ may have only one or two edges, thus the real performance would be better than the analytical performance $\mathcal{O}(\sum_{i=0}^{M}(m_{t_i})^{m_h} + |E|)$.

Algorithm 3 The exact algorithm for the temporal graph pattern.

Input: $G = (V, E)$: the temporal graph, $H = (g, \Delta T)$: the temporal graph pattern.
Output: C: the number of the subgraphs isomorphic to H in G.
1: $S \leftarrow$ partition the graph G into subgraphs $G_{t_i}, i = 0, 1..., M$;
2: **for** subgraph G_{t_i} in S **do**
3: $C_i \leftarrow$ count the number of the subgraphs isomorphic to H in G_{t_i};
4: **end for**
5: $C = \Sigma_{i=0}^{M} C_i$
6: **return** C

3.2. The Estimation Algorithm

In the previous subsection, we present the exact algorithm which has relatively high efficiency to reduce the intermediate results in the process of isomorphism. To further improve the computational efficiency, we design an algorithm based on the exact algorithm and the edge sampling in this subsection, where the edge sampling was used by Doulion [34] to count the triangles of the large graph. Instead of traversing all the edges in the graph, the algorithm here samples the edges by using the method of throwing biased coins. For each edge in the temporal graph, a biased coin with success probability p is thrown to determine whether this edge is reserved, which means that the selection probability of this edge is p. If the edge e is selected, e is placed in the edge set E_p; otherwise, e is ignored. Then, we build a new temporal graph G_p according to the selected edges E_p, and apply the exact algorithm to count the number C_p of isomorphic subgraphs in G_p. The approximate value of the exact number C can be obtained by C_p / p^{m_h}, where p^{m_h} is the probability of the TGP being selected. Since the algorithm outputs the estimated number of the isomorphic subgraphs, we call it the estimation algorithm. The specific steps of the algorithm is provided in Algorithm 4.

Algorithm 4 The estimation algorithm for the temporal graph pattern.

Input: $G = (V, E)$: the temporal graph, $H = (g, \Delta T)$: the temporal graph pattern, p: the probability of edge sampling.
Output: $\widehat{C}$: the approximate number of the temporal subgraphs isomorphic to H
1: $E_p = \emptyset$;
2: **for** all $e \in E$ **do**
3: $x \leftarrow$ Toss a biased coin with success probability p
4: **if** $x = 1$ **then**
5: $E_p \leftarrow E_p \cup e$
6: **end if**
7: **end for**
8: $G_p = (V_p, E_p) \leftarrow$ build a temporal graph according to the E_p,
9: $C_p \leftarrow$ count the number of H in G_p
10: $\widehat{C} = C_p / p^{m_h}$, where m_h is the number of the edges in H.
11: **return** $\widehat{C}$.

Although $\widehat{C}$ is an approximate value of C, $\widehat{C}$ is the unbiased estimation of C and has high accuracy (which can be seen in the numerical experiments). In the following, we give two lemmas about the expectation and variance of $\widehat{C}$ to demonstrate the unbiasedness.

Lemma 1. *Let C and $\widehat{C}$ be the exact number of the temporal subgraphs isomorphic to H in G and the estimated number obtained by Algorithm 4, respectively. Then, $E[\widehat{C}] = C$.*

Proof of Lemma 1. For each temporal subgraph isomorphic to H in G, we introduce an indicator variable $\delta_i, i = 1, ..., C$ to indicate whether the isomorphic subgraph is sampled or not, i.e.,

$$\delta_i = \begin{cases} 1, & if\ the\ subgraph\ is\ sampled \\ 0, & otherwise \end{cases}, i = 1, ..., C. \tag{1}$$

It is easy to see that the number C_p is obtained from the temporal graph after edge sampling, thus C_p is also the number of isomorphic subgraphs being sampled, i.e., $C_p = \sum_{i=1}^{C} \delta_i$. Then, we can get the expectation of $\widehat{C}$ as follows

$$E[\widehat{C}] = E\left[\frac{C_p}{p^{m_h}}\right] = E\left[\frac{1}{p^{m_h}} \sum_{i=1}^{C} \delta_i\right] = \frac{1}{p^{m_h}} \sum_{i=1}^{C} E[\delta_i] \tag{2}$$

where m_h is the number of the edges in H. Since each edge is sampled with probability p, the probability that the ith subgraph is selected (i.e., $\delta_i = 1$) is p^{m_h}. Then, the probability distribution of $\delta_i, i = 1, ...C$ can be denoted as.

According to Table 1, the expectation of δ_i can be calculated as

$$E[\delta_i] = 0 \cdot (1 - p^{m_h}) + 1 \cdot p^{m_h} = p^{m_h}. \tag{3}$$

By substituting Equation (3) into Equation (2), we can get $E[\widehat{C}] = C$. The proof is complete. □

Table 1. The probability distribution of $\delta_i, i = 1, ...C$.

δ_i	**0**	**1**
probability	$1 - p^{m_h}$	p^{m_h}

Lemma 2. *Let C and $\widehat{C}$ be the exact number of the temporal subgraphs isomorphic to H in G and the estimated number obtained by Algorithm 4, respectively. Then, $Var[\widehat{C}] = \frac{1}{p^{2m_h}} \left(\sum_{k=1}^{m_h} c_k (p^{2m_h - k} - p^{2m_h}) \right)$, where c_k is the number of cases that two subgraphs share k edges.*

Proof of Lemma 2. Similar to Equation (2), the variance $Var[\widehat{C}]$ is given by

$$Var[\widehat{C}] = Var\left[\frac{1}{p^{m_h}} \sum_{i=1}^{C} \delta_i\right] = \frac{1}{p^{2m_h}} \sum_{i=1}^{C} \sum_{j=1}^{C} Cov[\delta_i, \delta_j] \tag{4}$$

where $Cov[\delta_i, \delta_j]$ is the covariance of δ_i and δ_j, which can simplified as

$$Cov[\delta_i, \delta_j] = E[(\delta_i - E[\delta_i])(\delta_j - E[\delta_j])] = E[\delta_i \delta_j] - E[\delta_i]E[\delta_j] = E[\delta_i \delta_j] - p^{2m_h} \tag{5}$$

As can be seen from Equation (4), there are C^2 terms in the summation, where C terms belong to the case of $i = j$ and $C^2 - C$ terms belong to the case of $i \neq j$. To calculate the covariance $Cov[\delta_i, \delta_j]$, we divide the calculation into three cases.

- Case 1: $i = j$. When $i = j$, we have $Cov[\delta_i, \delta_j] = E[\delta_i^2] - p^{2m_h}$. Since δ_i obeys the 0–1 distribution shown in Table 1, $E[\delta_i^2]$ can be calculated as $E[\delta_i^2] = 0^2 \cdot (1 - p^{m_h}) + 1^2 \cdot p^{m_h} = p^{m_h}$. Then, we can get $Cov[\delta_i, \delta_j] = p^{m_h} - p^{2m_h}$.

- Case 2: $i \neq j$ and the subgraphs corresponding to δ_i, δ_j do not share any edges. In this case, whether the ith subgraph is sampled is unrelated to the jth subgraph, which means that δ_i and δ_j are independent. Thus, we have $Cov[\delta_i, \delta_j] = E[\delta_i]E[\delta_j] - p^{2m_h} = 0$.
- Case 3: $i \neq j$ and the subgraphs corresponding to δ_i, δ_j share edges. Assume that the ith and jth subgraphs share k ($k < m_h$) edges. Thus, the probability that the shared edges are sampled is p^k, and the probability that the remaining $2m_h - 2k$ edges of the two subgraphs are sampled is p^{2m_h-2k}. Based on these two probabilities, the probability of both subgraphs being sampled can be denoted as $E[\delta_i\delta_j] = p^k \cdot p^{2m_h-2k} = p^{2m_h-k}$. Then, we can obtain the covariance $Cov[\delta_i, \delta_j] = p^{2m_h-k} - p^{2m_h}$.

According to Equation (4) and the three cases, the variance $Var[\widehat{C}]$ can be rewritten as

$$\begin{aligned} Var[\widehat{C}] &= \frac{1}{p^{2m_h}}\left(C(p^{m_h} - p^{2m_h}) + c_0 \cdot 0 + \sum_{k=1}^{m_h-1} c_k(p^{2m_h-k} - p^{2m_h})\right) \\ &= \frac{1}{p^{2m_h}}\left(\sum_{k=1}^{m_h} c_k(p^{2m_h-k} - p^{2m_h})\right) \end{aligned} \tag{6}$$

where $c_k, k = 0, ..., m_h$ are the numbers of these terms $Cov(\delta_i, \delta_j), i, j = 1, ..., C$ in which the subgraphs represented by δ_i and δ_j share k edges, $c_{m_h} = C$ and $\sum_{k=0}^{m_h} c_k = C^2$. The proof is complete. □

It is worth noting that the estimation accuracy is closely related to the variance $Var[\widehat{C}]$. Actually, the expectation of the mean squared error (MSE) of $\widehat{C}$ is $E[(\widehat{C} - C)^2] = E[(\widehat{C} - E[\widehat{C}])^2] = Var[\widehat{C}]$. Moreover, compared to the exact algorithm, the estimation algorithm has higher computational efficiency. In the estimation algorithm, the edges in the temporal graph G are sampled, so the number of edges to be processed and the number of isomorphic subgraphs are reduced. Similar to the complexity of the exact algorithm (i.e., $\mathcal{O}(\sum_{i=0}^{M}(m_{t_i})^{m_h} + |E|)$), the computational complexity of the estimation algorithm can be expressed as $\mathcal{O}(\sum_{i=0}^{M'}(m'_{t_i})^{m_h} + p|E|)$, where m'_{t_i} is the edge number of G'_{t_i}, which is generated after partitioning the graph G_p, M' is the graph number after edge sampling and graph partitioning, and $\sum_{i=0}^{M'} m'_{t_i} = p|E|$. In the next section, we present several numerical experiments to verify that the algorithm has high estimation accuracy and efficiency.

4. Experiments and Discussion

4.1. Datasets and Setup

In this section, we evaluate the performance of the proposed algorithms in the following datasets.

CollegeMsg data [35]: The data record the private messages between users of the online social network at the University of California, Irvine. In these data, the temporal edge (u, v, t) of the temporal network means that the user u sends a private message to the user v at time t.

Email data [30]: These data were collected from European research institution and contain the e-mails between institution members from October 2003 to May 2005 (18 months). In these data, the directed edge (u, v, t) denotes that an e-mail is sent from member u to member v at time t.

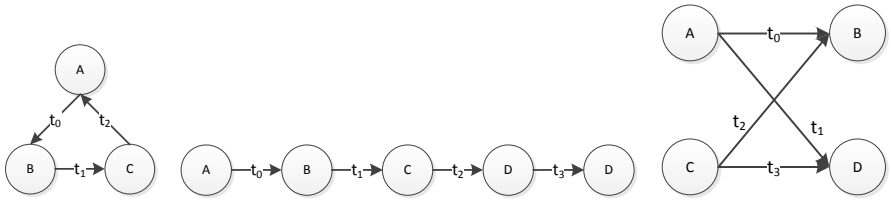
MathOverflow data [30]: The temporal network was generated using the interactions on the stack exchange website Math Overflow. The temporal edge (u, v, t) represents that user u answers user v's question or comments on user v's question/answer at time t.

Table 2 gives some statistics of these datasets. All experiments were conducted on the machine with Intel Core i7 3.40 GHz processor and 8 GB of memory. The software environment was Java 1.8.0.

Table 2. Statistics on the datasets.

Dataset	# Node	# Static Edges	# Temporal Edges	Time Span
CollegeMsg	1.9 K	20.3 K	59.8 K	194 days
Email	986	24.9 K	332 K	2.20 years
MathOverflow	24.8 K	228 K	390 K	6.44 years

To verify the performance of the proposed algorithms, we first chose the baseline algorithm. In fact, there are three works on the problem of counting TGP at present, i.e., the works in [30,32] and the BT algorithm [31]. However, the work in [30] only handles the motifs with at most three edges, and the sampling framework in [32] cannot be applied to our temporal motif definition. Therefore, the first two works cannot be used for the problem in this paper, thus we could only choose the BT algorithm [31] as the baseline algorithm. In the experiments, we re-implemented the code of the BT algorithm and made some modifications to make it suitable for our temporal motif definition. The temporal graph patterns (TGPs) used in the experiments are the typical motifs shown in Figure 1, where the topology of Triangle motif is rotational symmetric, and the topology of Bi-fan motif is axial symmetric.



(**a**) Triangle (**b**) Path (**c**) Bi-fan

Figure 1. Temporal graph patterns used in the experiments: (**a**) a classical triangle motif, and the temporal relationship is $t_0 < t_1 < t_2$; (**b**) s path motif whose edge time increases from left to right, and the motif indicates the propagation of information on the path; and (**c**) the bi-fan motif, and the temporal relationship is $t_0 < t_1 < t_2 < t_3$.

4.2. Experimental Results

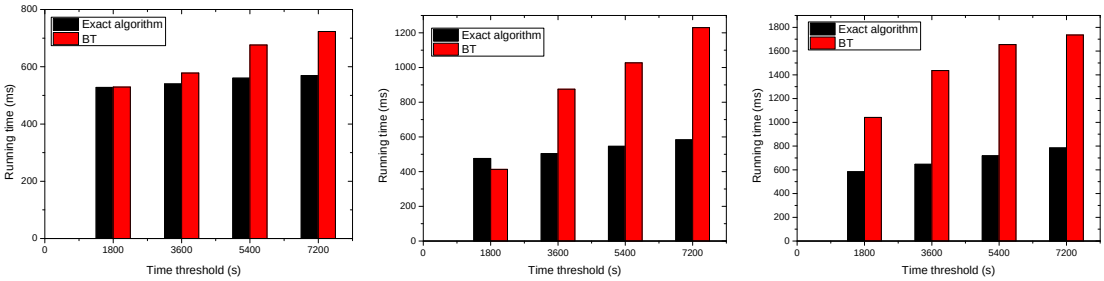
Next, we evaluated the performance of the exact algorithm and the estimation algorithm. First, we compared the performances of the exact algorithm and BT algorithm on the different datasets. Then, we defined two parameters to show the estimation accuracy and computational efficiency of the estimation algorithm.

4.2.1. Results of the Exact Algorithm

For the exact algorithm, the time threshold ΔT is an important parameter, thus we compared the performance under different ΔT. In this experiment, we applied the exact algorithm and the BT algorithm to count the above three TGPs in three datasets (CollegeMsg, Email and MathOverflow), and considered four different thresholds: 0.5 h (1800 s), 1 h (3600 s), 1.5 h (5400 s) and 2 h (7200 s). The results of these two algorithms are shown in Figures 2–4 and Table 3. In Table 3, the speedup ratio Sr is the ratio of running time of BT algorithm to that of exact algorithm. In these figures and table, we can get three findings as follows.

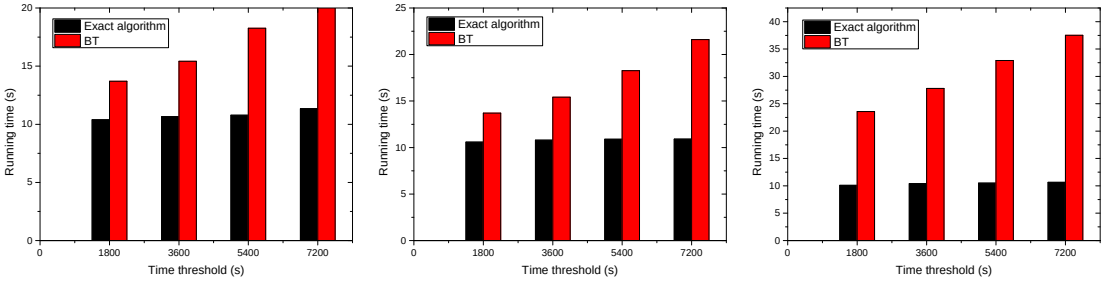
Table 3. The running time (s) and the speedup ratio of the exact and BT algorithms

Datasets	TGP	$\Delta T = 1800$			$\Delta T = 3600$			$\Delta T = 5400$			$\Delta T = 7200$		
		Exact	**BT**	***Sr***	**Exact**	**BT**	***Sr***	**Exact**	**BT**	***Sr***	**Exact**	**BT**	***Sr***
College	Triangle	0.52	0.52	1.0	0.54	0.57	1.1	0.56	0.67	1.2	0.56	0.723	1.3
	Path	0.47	0.41	0.8	0.50	0.87	1.7	0.54	1.02	1.9	0.58	1.23	2.1
	Bi-fan	0.58	1.04	1.8	0.64	1.43	2.2	0.71	1.65	2.3	0.78	1.73	2.2
Email	Triangle	10.39	11.24	1.1	10.65	12.42	1.2	10.78	13.70	1.3	11.34	16.22	1.4
	Path	10.58	13.69	1.3	10.79	15.41	1.4	10.88	18.26	1.7	10.91	21.59	2.0
	Bi-fan	10.10	23.56	2.3	10.40	27.79	2.7	10.50	32.90	3.1	10.65	37.52	3.5
Math	Triangle	11.00	10.64	0.9	11.25	11.12	1.0	12.28	12.35	1.0	12.82	13.62	1.1
	Path	10.19	10.61	1.0	11.23	11.27	1.0	12.59	13.48	1.1	12.79	14.34	1.1
	Bi-fan	12.29	21.91	1.8	13.47	22.62	1.7	13.50	23.78	1.8	13.40	25.63	1.9



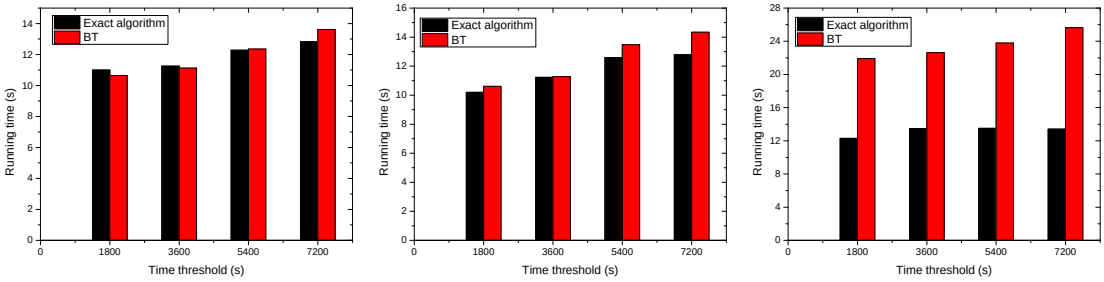
(**a**) Triangle (**b**) Path (**c**) Bi-fan

Figure 2. The running time of the exact and BT algorithms under different ΔT in CollegeMsg dataset. (**a**) the result for the Triangle TGP, (**b**) the result for the Path TGP, (**c**) the result for the Bi-fan TGP.



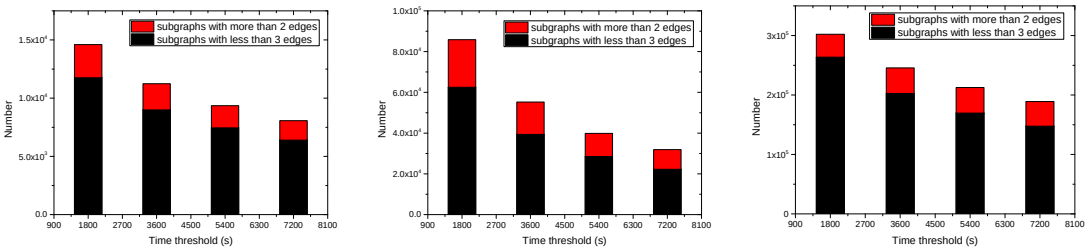
(**a**) Triangle (**b**) Path (**c**) Bi-fan

Figure 3. The running time of the exact and BT algorithms under different ΔT in Email dataset. (**a**) the result for the Triangle TGP, (**b**) the result for the Path TGP, (**c**) the result for the Bi-fan TGP.



(**a**) Triangle (**b**) Path (**c**) Bi-fan

Figure 4. The running time of the exact and BT algorithms under different ΔT in MathOverflow dataset. (**a**) the result for the Triangle TGP, (**b**) the result for the Path TGP, (**c**) the result for the Bi-fan TGP.



(**a**) CollegeMsg (**b**) Email (**c**) MathOverflow

Figure 5. The number of subgraphs in set S. (**a**) the result of CollegeMsg, (**b**) the result of Email, (**c**) the result of MathOverflow.

First, the running time of the two algorithms increases with the increase of the time threshold ΔT. To better explain the performance of the exact algorithm in Figures 2–4, Figure 5 gives the number of subgraphs in set S under different time thresholds. It can be seen that the number of subgraphs in set S decreases with the increase of ΔT. Therefore, the edge number of subgraph $G_{t_i}(G_{t_i} \in S)$ increases with the increase of ΔT. In Figure 5, we can also see that most of subgraphs in S have fewer than three edges, and the number of larger subgraphs decreases slightly with the increase of ΔT. Obviously, most of the reduced graph in S are the graphs with few edges and require very little calculation, which means that the decrease of the graph number in S has little impact on the performance of the algorithm. Thus, when the time threshold ΔT increases, the running time of the exact algorithm is mainly affected by the edge number of subgraph in S and increases inevitably. Similarly, with the increase of ΔT, the number of intermediate results processed by the BT algorithm increases, which results in the increase of the running time. Besides, it can also be seen in these figures that the change of running time caused by ΔT in BT algorithm is more obvious than that in the exact algorithm.

Moreover, the performance of the exact algorithm is more stable than that of the BT algorithm when counting different TGPs. From the results of different TGPs, we can see that the running time of the exact algorithm changes very little, which indicates that the algorithm performance is scarcely affected by the shape of TGP. In contrast, the BT algorithm has obvious increase in running time when dealing with the Bi-fan pattern, which is more complex than the other two patterns. Comparing the results in different datasets, it is easy to find that both algorithms spend less time on the CollegeMsg dataset, because this dataset has fewer temporal edges than the other two datasets.

Finally, the performance of the exact algorithm is significantly better than the performance of the BT algorithm. From theoretical complexity analysis above and the analysis in [31], we can see that the computational complexity of the exact algorithm is $\mathcal{O}(\sum_{i=0}^{M}(m_{t_i})^{m_h} + |E|)$, and that of the BT algorithm is $\mathcal{O}(|E|^{m_h})$, where m_{t_i}, m_h and $|E|$ are the edge numbers of G_{t_i}, H and G, respectively. It is easy to see that the computational complexity of the exact algorithm is lower than that of the BT algorithm. Although our algorithm is slightly slower than the BT algorithm in some cases, such as Figures 2b and 4a, the overall running time of our algorithm is better. Especially when ΔT is very large, the exact algorithm can even have a $3\times$ acceleration compared to the BT algorithm.

4.2.2. Results of the Estimation Algorithm

To evaluate the performance of the estimation algorithm, we defined the relative error ϵ and the speedup ratio Sr, where ϵ was used to measure the estimation accuracy and Sr was related to the computational efficiency. The definition of the ϵ is as follows:

$$\epsilon(C, \widehat{C}) = \frac{|\widehat{C} - C|}{C + 1} \tag{7}$$

In the above equation, we add 1 to both $\widehat{C}$ and C to avoid the case $C = 0$. The Sr is defined as

$$Sr = \frac{t}{t_e} \tag{8}$$

where t is the running time of the exact algorithm and t_e is running time of the estimation algorithm.

In this experiment, we tested the performance of the estimation algorithm under different probability values (0.6, 0.7, 0.8 and 0.9). Here, we only considered the case that the threshold ΔT is 7200 s, because the algorithm has similar characteristics using other time thresholds. Then, for each TGP and each probability, we repeated the proposed algorithms 10 times and computed the average ϵ and Sr.

Table 4 shows the relative error ϵ and the speedup ratio Sr of the estimation algorithm under different probabilities p. With the increase of the probability p, ϵ decreases from about 5% to 1%, which means that the estimation accuracy of the algorithm improves. Actually, the fluctuation of ϵ inevitably reduces when increasing p, because the larger is the p the smaller is the variance $Var[\widehat{C}]$ (as can be seen from Equation (6)). In this table, we can also see that the algorithm always has good estimation performance when dealing with different TGPs and datasets.

In addition, from the Sr in Table 4, we can see that the speedup ratio decreases with the increase of p, which means that the estimation algorithm takes more time when p is large. This is consistent with the fact that high probability p leads to high computational complexity of the estimation algorithm. Since the estimation algorithm still has high estimation accuracy when p is small, we can choose a relatively small p (e.g., 0.6–0.8) to achieve high accuracy and high computational efficiency at the same time.

Table 4. The average ϵ and Sr of the estimation algorithm under different probabilities (10 trials).

Datasets	TGP	$p = 0.6$		$p = 0.7$		$p = 0.8$		$p = 0.9$	
		ϵ	Sr	ϵ	Sr	ϵ	Sr	ϵ	Sr
CollegeMsg	Triangle	5.29%	2.00	3.93%	1.87	2.96%	1.62	1.43%	1.52
	Path	4.84%	2.23	3.02%	1.81	1.98%	1.44	1.67%	1.15
	Bi-fan	5.05%	2.20	2.95%	1.53	2.13%	1.48	1.98%	1.33
Email	Triangle	3.44%	2.92	1.35%	2.20	1.32%	1.71	1.17%	1.32
	Path	3.14%	2.85	2.52%	2.12	1.85%	1.64	1.01%	1.21
	Bi-fan	4.70%	2.74	2.24%	2.10	2.19%	1.61	1.19%	1.27
MathOverflow	Triangle	4.05%	3.03	2.85%	2.18	2.43%	1.64	1.47%	1.33
	Path	5.65%	2.88	4.43%	2.13	3.04%	1.59	1.96%	1.28
	Bi-fan	4.76%	2.94	3.76%	2.04	1.90%	1.53	1.56%	1.20

5. Discussion

In this section, we discuss the application scope and limitations of the algorithm.

(1) The type of network: In Section 3, we propose two algorithms for the counting problem in the static temporal network. Similar to all algorithms based on such network, the proposed algorithm cannot be applied to the dynamic network or the static network whose edges do not have the temporal information. It is worth noting that no algorithm can be applied to two different networks at the same time.

(2) The definition of TGP: In this paper, we consider the TGP defined in [27], which is different from the other definitions discussed in the Introduction. This definition is relatively suitable for communication networks [36], wikipedia network [28], and mobile cohesive groups [37].

(3) The algorithms: Since our algorithms partition the large graph into multiple subgraphs, the algorithms can be implemented in parallel, i.e., each subgraph is processed independently and the total result is obtained by adding the parallel results.

(4) The edge sampling: In the estimation algorithm, we use the edge sampling strategy to estimate the number of TGP. Since the increase in the edge number m_h of TGP reduces the probability of the

isomorphic subgraphs being sampled, the number of sampled isomorphic subgraphs will be small when m_h is large. Therefore, the algorithm has high relative error when counting the TGP with many edges. This indicates that it is only suitable for the TGP with a small number of edges. In the future, the problem of estimating the number of large TGP in the temporal graph can be further studied.

6. Conclusions

In this work, we propose an exact algorithm and an estimation algorithm for counting temporal graph patterns in large temporal graph. The exact algorithm is designed based on TFS. Since the algorithm can match the topology and the temporal relationship simultaneously, the high computational efficiency of the algorithm can be guaranteed. To further reduce the computational complexity, we then design the estimation algorithm based on the edge sampling. Extensive experiments on three real datasets showed that the exact algorithm is faster than the BT algorithm and the estimation algorithm can greatly reduce the running time while guaranteeing the estimation accuracy.

Author Contributions: Conceptualization, X.S. and Y.T.; methodology, X.S.; software, X.S. and J.W.; validation, X.S. and Y.T.; formal analysis, X.S. and Q.W.; investigation, Y.T., Q.W., and C.S.; resources, Y.T. and C.S.; writing—original draft preparation, X.S.; writing—review and editing, Y.T. and Q.W.; project administration, Q.W.; and funding acquisition, Y.T.

Funding: This work was funded by the National Natural Science Foundation of China under grant No. 61872444 and the National Key Research and Development Program of China under grant No. 2018YFB1003602.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gao, W.; Wu, H.; Siddiqui, M.K.; Baig, A.Q. Study of biological networks using graph theory. *Saudi J. Biol. Sci.* **2018**, *25*, 1212–1219. [CrossRef] [PubMed]
2. Trinajstic, N. *Chemical Graph Theory*; Routledge: London, UK, 1992.
3. Yang, P.; Freeman, R.A.; Gordon, G.J.; Lynch, K.M.; Srinivasa, S.S.; Sukthankar, R. Decentralized estimation and control of graph connectivity for mobile sensor networks. *Automatica* **2010**, *46*, 390–396. [CrossRef]
4. Yu, J.; Shen, Y.; Yang, Z. Topic-STG: Extending the session-based temporal graph approach for personalized tweet recommendation. In Proceedings of the 23rd International Conference on World Wide Web, Seoul, Korea, 7–11 April 2014; ACM: New York, NY, USA, 2014; pp. 413–414.
5. Choudhury, S.; Holder, L.; Chin, G.; Ray, A.; Beus, S.; Feo, J. Streamworks: A system for dynamic graph search. In Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 22–27 June 2013; pp. 1101–1104.
6. Kovanen, L.; Kaski, K.; Kertész, J.; Saramäki, J. Temporal motifs reveal homophily, gender-specific patterns, and group talk in call sequences. *Proc. Natl. Acad. Sci. USA* **2013**, *110*, 18070–18075. [CrossRef] [PubMed]
7. Harshaw, C.R.; Bridges, R.A.; Iannacone, M.D.; Reed, J.W.; Goodall, J.R. Graphprints: Towards a graph analytic method for network anomaly detection. In Proceedings of the 11th Annual Cyber and Information Security Research Conference, Oak Ridge, TN, USA, 5–7 April 2016; p. 15.
8. Berry, J.W.; Hendrickson, B.; LaViolette, R.A.; Phillips, C.A. Tolerating the community detection resolution limit with edge weighting. *Phys. Rev. E* **2011**, *83*, 056119. [CrossRef] [PubMed]
9. Iliofotou, M.; Faloutsos, M.; Mitzenmacher, M. Exploiting dynamicity in graph-based traffic analysis: Techniques and applications. In Proceedings of the 5th international conference on Emerging networking experiments and technologies, Rome, Italy, 1–4 December 2009; pp. 241–252.
10. Braha, D.; Bar-Yam, Y. Time-Dependent Complex Networks: Dynamic Centrality, Dynamic Motifs, and Cycles of Social Interactions. In *Adaptive Networks: Theory, Models and Applications*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 39–50.
11. Al Hasan, M.; Dave, V.S. Triangle counting in large networks: A review. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1226. [CrossRef]
12. Lim, Y.; Kang, U. Mascot: Memory-efficient and accurate sampling for counting local triangles in graph streams. In Proceedings of the 21th ACM SIGKDD International Conference On Knowledge Discovery and Data Mining, Sydney, NSW, Australia, 10–13 August 2015; pp. 685–694.

13. Pearce, R. Triangle counting for scale-free graphs at scale in distributed memory. In Proceedings of the 2017 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 12–14 September 2017; pp. 1–4.
14. Stefani, L.D.; Epasto, A.; Riondato, M.; Upfal, E. TRIÈST: Counting Local and Global Triangles in Fully-dynamic Streams with Fixed Memory Size. *ACM Trans. Knowl. Discov. Data (TKDD)* **2017**, *11*, 43. [CrossRef]
15. Pinar, A.; Seshadhri, C.; Vishal, V. Escape: Efficiently counting all 5-vertex subgraphs. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 1431–1440.
16. Marcus, D.; Shavitt, Y. Efficient counting of network motifs. In Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems Workshops, Genova, Italy, 21–25 June 2010; pp. 92–98.
17. Hočevar, T.; Demšar, J. A combinatorial approach to graphlet counting. *Bioinformatics* **2014**, *30*, 559–565. [CrossRef] [PubMed]
18. Wang, P.; Zhao, J.; Zhang, X.; Li, Z.; Cheng, J.; Lui, J.C.; Towsley, D.; Tao, J.; Guan, X. MOSS-5: A fast method of approximating counts of 5-node graphlets in large graphs. *IEEE Trans. Knowl. Data Eng.* **2017**, *30*, 73–86. [CrossRef]
19. Chen, X.; Lui, J.C.S. Mining Graphlet Counts in Online Social Networks. *ACM Trans. Knowl. Discov. Data (TKDD)* **2018**, *12*, 41:1–41:38. [CrossRef]
20. Bressan, M.; Chierichetti, F.; Kumar, R.; Leucci, S.; Panconesi, A. Counting graphlets: Space vs. time. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, Cambridge, UK, 6–10 February 2017; pp. 557–566.
21. Bhuiyan, M.A.; Rahman, M.; Rahman, M.; Al Hasan, M. Guise: Uniform sampling of graphlets for large graph analysis. In Proceedings of the 12th International IEEE Conference on Data Mining, Brussels, Belgium, 10–13 December 2012; pp. 91–100.
22. Rahman, M.; Bhuiyan, M.A.; Al Hasan, M. Graft: An Efficient Graphlet Counting Method for Large Graph Analysis. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 2466–2478. [CrossRef]
23. Jha, M.; Seshadhri, C.; Pinar, A. Path Sampling: A Fast and Provable Method for Estimating 4-Vertex Subgraph Counts. In Proceedings of the 24th International Conference on World Wide Web, 2015, WWW '15, Florence, Italy, 18–22 May 2015; pp. 495–505.
24. Bressan, M.; Leucci, S.; Panconesi, A. Motivo: Fast motif counting via succinct color coding and adaptive sampling. *arXiv* **2019**, arXiv:1906.01599.
25. Bajardi, P.; Barrat, A.; Natale, F.; Savini, L.; Colizza, V. Dynamical patterns of cattle trade movements. *PLoS ONE* **2011**, *6*, e19869. [CrossRef] [PubMed]
26. Zhao, Q.; Tian, Y.; He, Q.; Oliver, N.; Jin, R.; Lee, W.C. Communication motifs: A tool to characterize social communications. In Proceedings of the 19th ACM International Conference on Information and Knowledge Management, Toronto, ON, Canada, 26–30 October 2010; pp. 1645–1648.
27. Kovanen, L.; Karsai, M.; Kaski, K.; Kertész, J.; Saramäki, J. Temporal motifs in time-dependent networks. *J. Stat. Mech. Theory Exp.* **2011**, *2011*, P11005. [CrossRef]
28. Jurgens, D.; Lu, T.C. Temporal Motifs Reveal the Dynamics of Editor Interactions in Wikipedia. In Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media (ICWSM 2012), Dublin, Ireland, 4–7 June 2012.
29. Wu, W.; Hu, X.; Guo, S.; He, X. Methods of Analyzing Combat SoS Coordination Pattern Based on Temporal Motif. In *Theory, Methodology, Tools and Applications for Modeling and Simulation of Complex Systems*; Springer: Singapore, 2016; pp. 544–554.
30. Paranjape, A.; Benson, A.R.; Leskovec, J. Motifs in Temporal Networks. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, Cambridge, UK, 6–10 February 2017; pp. 601–610.
31. Mackey, P.; Porterfield, K.; Fitzhenry, E.; Choudhury, S.; Chin, G. A chronological edge-driven approach to temporal subgraph isomorphism. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 3972–3979.
32. Liu, P.; Benson, A.R.; Charikar, M. Sampling methods for counting temporal motifs. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, Melbourne VIC, Australia, 11–15 February 2019; pp. 294–302.
33. Sun, X.; Tan, Y.; Wu, Q.; Chen, B.; Shen, C. TM-Miner: TFS-Based Algorithm for Mining Temporal Motifs in Large Temporal Network. *IEEE Access* **2019**, *7*, 49778–49789. [CrossRef]

34. Tsourakakis, C.E.; Kang, U.; Miller, G.L.; Faloutsos, C. Doulion: Counting triangles in massive graphs with a coin. In Proceedings of the 15th ACM SIGKDD International Conference On Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; pp. 837–846.
35. Panzarasa, P.; Opsahl, T.; Carley, K.M. Patterns and Dynamics of Users' Behavior and Interaction: Network Analysis of an Online Community. *J. Assoc. Inf. Sci. Technol.* **2010**, *60*, 911–932. [CrossRef]
36. Gurukar, S.; Ranu, S.; Ravindran, B. COMMIT: A Scalable Approach to Mining Communication Motifs from Dynamic Networks. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, 31 May–4 June 2015; pp. 475–489.
37. Zignani, M.; Quadri, C.; Del Vicario, M.; Gaito, S.; Rossi, G.P. Temporal Communication Motifs in Mobile Cohesive Groups. In Proceedings of the International Workshop on Complex Networks and Their Applications, Cambridge, UK, 11–13 December 2018; pp. 490–501.