Time-Topology Analysis

Yunkai Lou[1], Chaokun Wang[1], Tiankai Gu[1], Hao Feng[1], Jun Chen[2], Jeffrey Xu Yu[3]
[1] School of Software, Tsinghua University, Beijing 100084, China
[2] Baidu Inc., Beijing, China
[3] The Chinese University of Hong Kong, Hong Kong, China
{louyk18,gtk18,fh20}@mails.tsinghua.edu.cn,chaokun@tsinghua.edu.cn,chenjun22@baidu.com,yu@se.cuhk.edu.hk

ABSTRACT

Many real-world networks have been evolving, and are finely modeled as temporal graphs from the viewpoint of the graph theory. A temporal graph is informative, and always contains two types of information, i.e., the temporal information and topological information, where the temporal information reflects the time when the relationships are established, and the topological information focuses on the structure of the graph. In this paper, we perform time-topology analysis on temporal graphs to extract useful information. Firstly, a new metric named $\mathbb{T}$-cohesiveness is proposed to evaluate the cohesiveness of a temporal subgraph. It defines the cohesiveness of a temporal subgraph from the time and topology dimensions jointly. Specifically, given a temporal graph $\mathcal{G}_s = (V_s, \mathcal{E}_s)$, cohesiveness in the time dimension reflects whether the connections in $\mathcal{G}_s$ happen in a short period of time, while cohesiveness in the topology dimension indicates whether the vertices in V_s are densely connected and have few connections with vertices out of $\mathcal{G}_s$. Then, $\mathbb{T}$-cohesiveness is utilized to perform time-topology analysis on temporal graphs, and two time-topology analysis methods are proposed. In detail, $\mathbb{T}$-cohesiveness evolution tracking traces the evolution of the $\mathbb{T}$-cohesiveness of a subgraph, and combo searching finds out all the subgraphs that contain the query vertex and have $\mathbb{T}$-cohesiveness larger than a given threshold. Moreover, a pruning strategy is proposed to improve the efficiency of combo searching. Experimental results confirm the efficiency of the proposed time-topology analysis methods and the pruning strategy.

PVLDB Reference Format:
Yunkai Lou, Chaokun Wang, Tiankai Gu, Hao Feng, Jun Chen, Jeffrey Xu Yu. Time-Topology Analysis. PVLDB, 14(13): 3322 - 3334, 2021.
doi:10.14778/3484224.3484230

1 INTRODUCTION

It is well known that real-world networks, from online social networks to protein-protein interaction networks, change with time permanently. Usually, these evolving networks are finely modeled as temporal graphs from the viewpoint of graph theory [17, 25]. A temporal graph consists of a set of vertices and temporal edges among them, and always contains two types of information, i.e., the temporal information and topological information, where the former reflects the time when the relationships are established [25], and the latter is the relationships among the vertices. Then, it is of great significance to analyze the temporal graphs, so that noteworthy information and substructures can be extracted.

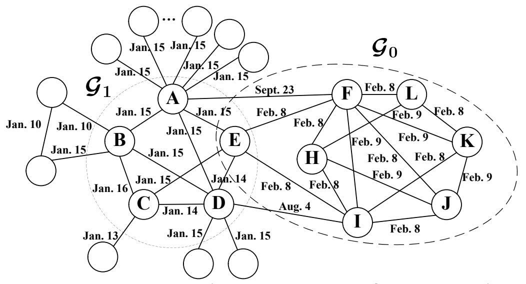


Figure 1: A sample money transfer network.

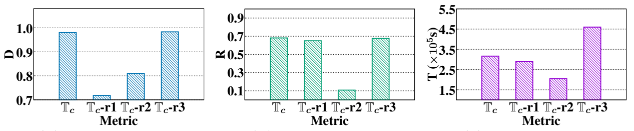
There have been many methods for network/graph analysis, and they can be mainly divided into three classes: For static graphs, various community detection [2, 14, 23, 32, 40] and community search [19, 21, 26, 28, 43] algorithms are proposed to find dense subgraphs; For attributed graphs, algorithms are proposed to find cohesive subgraphs according to the network structure and vertex attributes [8, 20, 41, 44, 45]; For temporal graphs, existing methods [11, 37, 46] always break down the graphs into snapshots first, and then find dense subgraphs in the snapshots. However, these existing methods never properly utilize the temporal information, and are poor at discovering some interesting patterns in temporal graphs.

Example 1. *Given a money transfer network shown in Figure 1, the vertices and edges represent the accounts and the transactions among them, respectively. There is a timestamp on each temporal edge representing the start time of the corresponding transaction. A group of accounts are cohesive, if they satisfy that (1) many transactions occur among these accounts (density); (2) these accounts have few relationships with the accounts out of the group (cohesion); (3) these transactions occur in a short period of time (short-time). The obtained cohesive accounts can help in many practical applications such as money laundering detection [24] and risk management [31].*

To find cohesive account groups with a given account (saying vertex E), the existing methods usually have poor performances, since they often utilize the temporal information improperly. For instance, based on BZ [21], one of the state-of-the-art community search methods, the induced subgraph of vertices A, B, C, D, E, F, H, I, J, K, and L is returned (a kind of k-core structures). However, these accounts are actually not cohesive, because the occurrence time of the transactions among them differs greatly.

One intuitive idea is to partition the temporal network along the time dimension by a sliding window with the step length 1 (i.e., one minimum time unit), and then search for communities in each temporal graph within a time window (e.g., with a width of 30 time units).

Proceedings of the VLDB Endowment, Vol. 14, No. 13 ISSN 2150-8097.
doi:10.14778/3484224.3484230



(a) Average Ds (b) Average Rs (c) Average Ts

Figure 2: The result of one ablation study on *Col-7d*. Given a temporal graph $\mathcal{G}$, for a cohesive subgraph $\mathcal{G}_s = (V_s, \mathcal{E}_s)$ of $\mathcal{G}$, D is $\mathcal{G}_s$'s density, R is the ratio of $|\mathcal{E}_s|$ to the number of temporal edges (in $\mathcal{G}$) adjacent to V_s, and T is $\mathcal{G}_s$'s time span.

However, the number of time windows is so large that the overall time cost of community search is prohibitive.

Furthermore, in a given time window, it is not easy to detect ideal communities. With the time window of [Jan. 14, Jan. 16], the induced subgraph of vertices A, B, C, D, and E is returned by BZ, and satisfies Requirement 1 (density). Clearly, however, it violates Requirement 2 (cohesion), i.e. having more relationships with accounts outside the community. In detail, plenty of transactions are related with Account A (indeed, A is an account of a company that pays its employees on Jan. 15), which indicates that A is not cohesive with B, C, D, and E. In other words, BZ does not consider the density and the cohesion together. Similar are the other existing methods.

The ideal result in Figure 1 is the induced subgraph of vertices E, F, H, I, J, K, and L, i.e., $\mathcal{G}_0$. In order to find out such temporal subgraphs (i.e. cohesive subgraphs) effectively, where vertices are closely related in both the time and topology dimensions, we devise a new technique named time-topology analysis.

In our time-topology analysis, the above three requirements are all considered. The vertices in a qualified temporal subgraph should be densely connected (Requirement 1) and have few connections with vertices out of the subgraph (Requirement 2). Besides, the edges in the subgraph should have close timestamps (Requirement 3). Clearly, as shown in Figure 1, the ideal subgraph cannot be found just according to one of the requirements. For example, based on Requirement 1 (saying k-core is used), the resultant subgraph is too large and has a long time span if $k = 3$, while there is no resultant subgraph containing Vertex E if $k = 4$. Then, Requirements 2 and 3 should also be considered to achieve the satisfactory result.

Next, given a temporal subgraph $\mathcal{G}_s$, three probabilities (i.e., ζ_{intra}, ζ_{inter}, and ζ_t) are defined to measure the cohesiveness of $\mathcal{G}_s$ from the viewpoints of these three requirements, separately. Then, the overall probability of $\mathcal{G}_s$ being cohesive (denoted as $\mathbb{T}$-cohesiveness, abbr. $\mathbb{T}_c$) is proposed, which is the multiplication of the three probabilities and measures the whole cohesiveness of $\mathcal{G}_s$. As shown in Figure 2, the results of an ablation study demonstrate that every requirement is important. Specifically, $\mathbb{T}_c$ (the first bars in Figure 2a–2c) represents the method we propose, and it considers all the three requirements. $\mathbb{T}_c$-r1 (the second bars), $\mathbb{T}_c$-r2 (the third bars), and $\mathbb{T}_c$-r3 (the fourth bars) are the methods ignoring ζ_{intra} (Requirement 1), ζ_{inter} (Requirement 2), and ζ_t (Requirement 3) , respectively. When $\mathbb{T}_c$-r1 is used to find cohesive subgraphs, the obtained subgraphs have small values of D as shown in Figure 2a, and are sparse as well as not closely connected. When $\mathbb{T}_c$-r2 is used, the obtained subgraphs have quite small values of R ($R \approx 0.1$) as shown in Figure 2b, and they have much more connections with vertices out of the subgraphs. When $\mathbb{T}_c$-r3 is used, the obtained subgraphs have much larger Ts (shown in Figure 2c), which indicates that they have much longer time spans and are not cohesive in the time dimension. More details are given in Section 6.5 of the technical report w.r.t. this paper [27]. Then, $\mathbb{T}_c$ is used in time-topology analysis to evaluate the cohesiveness of temporal subgraphs.

Table 1: Notations used in this paper.

Notation	Description
$\mathcal{G} = (V, \mathcal{E})$	a temporal graph with vertex set V and temporal edge set $\mathcal{E}$
$I_{\mathcal{G}}(V_s)$	the induced subgraph of vertex group V_s on $\mathcal{G}$
$G^P = (V^P, E^P)$	a projected graph
a combo $\mathcal{G}_s = (V_s, \mathcal{E}_s)$	an induced subgraph of V_s on $\mathcal{G}$ satisfying the constraints of temporal cohesiveness and topological cohesiveness
$\mathcal{G}_s^l = (V_s^l, \mathcal{E}_s^l)$	the local structure of vertex group V_s^l w.r.t. $\mathcal{G}$
$N(V_s)$	the neighbors of vertices in V_s
$T(\mathcal{G}_s)$	the time span of $\mathcal{G}_s$, abbr. T
$R(\mathcal{G}_s)$	$R(\mathcal{G}_s) = \lvert\mathcal{E}_s\rvert / \lvert\mathcal{E}_s^l\rvert$, the ratio of the number of temporal edges in $\mathcal{G}_s$ to that of temporal edges adjacent to vertices in V_s, abbr. R
$D(\mathcal{G}_s)$	the density of $\mathcal{G}_s$, abbr. D
$\zeta_t(\mathcal{G}_s), \zeta_{\text{inter}}(\mathcal{G}_s), \zeta_{\text{intra}}(\mathcal{G}_s)$	the TC score, InterTC score, and IntraTC score of $\mathcal{G}_s$
$\mathbb{T}_c(\mathcal{G}_s)$	$\mathbb{T}_c = \zeta_t(\mathcal{G}_s) * \zeta_{\text{inter}}(\mathcal{G}_s) * \zeta_{\text{intra}}(\mathcal{G}_s)$, the $\mathbb{T}$-cohesiveness of $\mathcal{G}_s$
$\widehat{\zeta}_t, \widehat{\zeta}_{\text{inter}}, \widehat{\zeta}_{\text{intra}}$	the upper bounds of $\zeta_t, \zeta_{\text{inter}}$ and ζ_{intra}

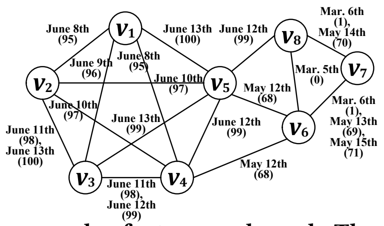


Figure 3: An example of a temporal graph. The date on each edge represents the day the interaction happens, and the number in the parenthesis is the corresponding timestamp.

The main contributions of this paper are summarized as follows:

(1) We are the first to perform time-topology analysis on temporal graphs to the best of our knowledge. In the process of such analysis, both temporal and topological information are considered.

(2) Some useful concepts are presented (in Section 2), and a new evaluation metric named Temporal-and-Topological Cohesiveness (abbr. $\mathbb{T}$-cohesiveness) is proposed to evaluate the cohesiveness of temporal subgraphs (in Section 3). $\mathbb{T}$-cohesiveness takes both the temporal and topological cohesiveness of a temporal subgraph into account, and is utilized to perform time-topology analysis.

(3) Two time-topology analysis methods are proposed based on $\mathbb{T}$-cohesiveness, i.e., $\mathbb{T}$-cohesiveness evolution tracking and combo searching (in Section 4). Then, an optimization method is designed to improve the efficiency of combo searching (in Section 5).

(4) Experimental results (in Section 6) confirm the efficiency and good scalability of our proposed methods. The ablation study shows the good performance of the proposed optimization method.

Because of the length limitation, some algorithms, proofs, and experiments are omitted here, and detailed in [27].

2 PRELIMINARY

In this section, some necessary concepts are presented, and the notations used in this paper are summarized in Table 1.

Definition 1 (Temporal Graph). *A temporal graph is denoted as $\mathcal{G} = (V, \mathcal{E})$, where V is the set of vertices, and $\mathcal{E}$ is the set of* *temporal edges. Each temporal edge $e \in \mathcal{E}$ is a triplet (u, v, t), where $u, v \in V$, and t is the timestamp that stores the interaction time of u and v. The minimal and maximal timestamps in $\mathcal{G}$ are denoted as t_{min} and t_{max} respectively, and the time span of $\mathcal{G}$ is $t_{max} - t_{min}$. A subgraph of $\mathcal{G}$ is called a temporal subgraph.*

Example 2. *Figure 3 is a temporal graph. For instance, the temporal edges between v_2 and v_3 are denoted as $(v_2, v_3, 98)$ and $(v_2, v_3, 100)$. Besides, we have $t_{min} = 0$ and $t_{max} = 100$ for this graph.*

Definition 2 (Induced Subgraph). *Given a temporal graph $\mathcal{G} = (V, \mathcal{E})$ and a vertex group V_s, the induced subgraph of V_s on $\mathcal{G}$ is $I_{\mathcal{G}}(V_s) = (V_s, \mathcal{E}_s)$, where $\mathcal{E}_s = \{(u, v, t) | u, v \in V_s, (u, v, t) \in \mathcal{E}\}$. It can be abbreviated as $I(V_s)$ when there is no ambiguity.*

If the temporal information is neglected, a temporal graph reduces to a projected graph. The definition of a projected graph is as follows.

Definition 3 (Projected Graph). *Given a temporal graph $\mathcal{G} = (V, \mathcal{E})$, the projected graph of $\mathcal{G}$ is $G^P = (V^P, E^P)$, where $E^P = \{(u, v) | (u, v, t) \in \mathcal{E}\}$. Each $(u, v) \in E^P$ is called a normal edge.*

Definition 4 (Combo). *Given a temporal graph $\mathcal{G}$ and a vertex group V_s, $\mathcal{G}_s = I_{\mathcal{G}}(V_s)$ is called a combo iff it satisfies: (1)* ***Temporal Cohesiveness:*** *The timestamps of temporal edges in $\mathcal{G}_s$ are in a short period of time; (2)* ***Topological Cohesiveness:*** *Vertices in $\mathcal{G}_s$ are densely connected, and there are few temporal edges between a vertex in $\mathcal{G}_s$ and a vertex out of $\mathcal{G}_s$. Besides, V_s is said to form the combo.*

For simplicity, given a subgraph $\mathcal{G}_s$, the temporal edges in $\mathcal{G}_s$ are called the intra-edges, and the temporal edges between a vertex in $\mathcal{G}_s$ and a vertex out of $\mathcal{G}_s$ are called the inter-edges. Note that the subgraphs in this paper are induced subgraphs if not specified.

Definition 5 (Local Structure). *Given a temporal graph $\mathcal{G} = (V, \mathcal{E})$ and a vertex group V_s, the local structure of V_s w.r.t. $\mathcal{G}$, which is denoted as $\mathcal{G}_s^l = (V_s^l, \mathcal{E}_s^l)$, satisfies that $V_s^l = V_s \cup N(V_s)$ and $\mathcal{E}_s^l$ consists of all the temporal edges adjacent to vertices in V_s, where $N(V_s)$ represents the set of the neighbors of the vertices in V_s.*

3 $\mathbb{T}$-COHESIVENESS

In this section, the main idea of Temporal-and-Topological Cohesiveness ($\mathbb{T}$-cohesiveness) is proposed, the three terms of $\mathbb{T}$-cohesiveness, i.e., ζ_t, ζ_{inter}, and ζ_{intra}, are detailed, and the method to compute the $\mathbb{T}$-cohesiveness of a temporal subgraph is explained.

3.1 Main Idea of $\mathbb{T}$-cohesiveness

As $\mathbb{T}$-cohesiveness is designed for evaluating the cohesiveness of a temporal subgraph or combo $\mathcal{G}_s$, it should consider both the temporal and topological information of $\mathcal{G}_s$. Firstly, the vertices in a combo should be densely connected, and there should be few edges between a vertex in the combo and a vertex out of the combo. Secondly, the creation time of the temporal edges in a combo should be within a short period of time. Based on these considerations, we propose a novel evaluation metric named temporal-and-topological cohesiveness (abbr. $\mathbb{T}$-cohesiveness) to evaluate temporal subgraphs.

Specifically, given a temporal graph $\mathcal{G} = (V, \mathcal{E})$ and a temporal subgraph $\mathcal{G}_s = (V_s, \mathcal{E}_s)$ of $\mathcal{G}$, the $\mathbb{T}$-cohesiveness of $\mathcal{G}_s$, denoted as $\mathbb{T}_c(\mathcal{G}_s)$, is defined as follows:

$$\mathbb{T}_c(\mathcal{G}_s) = \zeta_t(\mathcal{G}_s) * \zeta_{\text{inter}}(\mathcal{G}_s) * \zeta_{\text{intra}}(\mathcal{G}_s) \tag{1}$$

Equation 1 consists of three terms multiplied together to obtain the $\mathbb{T}$-cohesiveness of $\mathcal{G}_s$. The first term ζ_t is called the temporal cohesiveness score. It reflects the temporal cohesiveness of $\mathcal{G}_s$, and is in the range of (0, 1]. The second and third terms are called the inter-topological cohesiveness score and intra-topological cohesiveness score respectively. They reflect the topological cohesiveness of $\mathcal{G}_s$, and their values are in the range of [0, 1]. Specifically, ζ_{inter} reflects how much the intra-edges of $\mathcal{G}_s$ are more than the inter-edges, while ζ_{intra} focuses on how densely the vertices in $\mathcal{G}_s$ are connected. The three terms can be considered as the probabilities of $\mathcal{G}_s$ being cohesive from three perspectives (temporal cohesiveness, inter-topological cohesiveness, and intra-topological cohesiveness), respectively, and the value of $\mathbb{T}$-cohesiveness is the overall probability that $\mathcal{G}_s$ is cohesive. These three terms are multiplied rather than summed in the form of $a\zeta_t + b\zeta_{\text{inter}} + c\zeta_{\text{intra}}$ mainly because a temporal subgraph with a quite small ζ_t, ζ_{inter}, or ζ_{intra} is not cohesive, and then should have a quite small value of $\mathbb{T}$-cohesiveness.

Then, the value of $\mathbb{T}_c$ is in the range of [0, 1], and a larger $\mathbb{T}_c$ indicates a more cohesive subgraph. Note that the $\mathbb{T}$-cohesiveness of a vertex group V_s means the $\mathbb{T}$-cohesiveness of $I_{\mathcal{G}}(V_s)$. The details of ζ_t, ζ_{inter}, and ζ_{intra} are presented in the following subsections.

3.2 Temporal Cohesiveness Score ζ_t

As proposed in Definition 4, the temporal cohesiveness of a subgraph reflects whether the edges in the subgraph are established in a short period of time. In this subsection, the temporal cohesiveness score ζ_t (abbr. the TC score) is presented to measure the temporal cohesiveness of a temporal subgraph. Specifically, ζ_t of a subgraph is defined based on the time span of the subgraph as follows:

$$\zeta_t(\mathcal{G}_s) = \frac{1}{1 + log\left(\frac{e-1}{T_{0.5}-T_1} * \left(\max(T(\mathcal{G}_s), T_1) - T_1\right) + 1\right)} \tag{2}$$

where $T(\mathcal{G}_s)$ is the time span of $\mathcal{G}_s$, and can be abbreviated as T when there is no ambiguity. T_1 and $T_{0.5}$ are user-specified. In detail, temporal subgraphs with time spans T_1 and $T_{0.5}$ are set to have the TC scores of 1 and 0.5 respectively in Equation 2. Besides, $\max(T, T_1)$ is used to ensure that a subgraph with a time span shorter than T_1 has a TC score of 1. $\frac{e-1}{T_{0.5}-T_1}$ is used to ensure that $\zeta_t(\mathcal{G}_s) = 0.5$ when $T = T_{0.5}$. Note that a shorter time span indicates a more temporally-cohesive subgraph, and contributes to a larger ζ_t. ζ_t is defined in the form of $\frac{1}{1+log(x)}$ mainly for two reasons: (1) The scoring function should have better distinguishing ability when the time span is near T_1. Therefore, its (absolute) gradient should increase with the decrease of T. (2) Equation 2 in the form of $\frac{1}{1+log(x)}$ has larger (absolute) gradient than $\frac{1}{1+\frac{\max(T,T_1)-T_1}{T_{0.5}-T_1}}$ in the form of $\frac{1}{1+x}$ when T is near T_1. The reasons for defining ζ_{inter} and ζ_{intra} in the form of $\frac{1}{1+log(x)}$ are similar and omitted for brevity.

Leaving two factors (i.e., T_1 and $T_{0.5}$) adjustable can make ζ_t adaptive to different conditions. For example, in a coauthorship graph, some researchers who coauthored papers only in a specific year should be considered temporally cohesive, and we can set $T_1 =$ 0 year. However, in a temporal graph representing money transfer, a temporally cohesive subgraph should have a much shorter time span such as a week. Then, we can set $T_1 = 6$ days to evaluate the temporal cohesiveness of subgraphs in this temporal graph.

Example 3. *We set $T_1 = 4$ and $T_{0.5} = 7$, which means that if the persons in a temporal subgraph contact in a time span of four days, the TC score is 1, and if they contact in a time span of seven days, the TC score is 0.5. As shown in Figure 3, let $\mathcal{G}_s = I(\{v_1, v_2, v_3, v_4, v_5\})$. Then, the time span of $\mathcal{G}_s$ is five days, and $\zeta_t(\mathcal{G}_s) = 0.69$.*

3.3 Inter-Topological Cohesiveness Score ζ_{inter}

Inter-topological cohesiveness score ζ_{inter} (abbr. the InterTC score) focuses on the number of intra-edges compared with the inter-edges. Specifically, if a temporal subgraph $\mathcal{G}_s = (V_s, \mathcal{E}_s)$ has much more intra-edges than inter-edges, most of the temporal edges adjacent to its vertices are in $\mathcal{G}_s$ itself, and $\mathcal{G}_s$ has a large InterTC score. The ratio of the intra-edges in $\mathcal{G}_s = (V_s, \mathcal{E}_s)$ is computed as follows:

$$R(\mathcal{G}_s) = \frac{|\mathcal{E}_s|}{|\mathcal{E}_s^l|} \tag{3}$$

where $|\mathcal{E}_s^l|$ represents the number of temporal edges in the local structure of $\mathcal{G}_s$. $R(\mathcal{G}_s)$ is abbreviated as R when there is no ambiguity. Then, the InterTC score of $\mathcal{G}_s$ is defined as follows:

$$\zeta_{\text{inter}}(\mathcal{G}_s) = \begin{cases} \frac{1}{1+log\left(\frac{e-1}{R_1-R_{0.5}}*(R_1-\min(R(\mathcal{G}_s),R_1))+1\right)} & |\mathcal{E}_s| \neq 0 \\ 0 & |\mathcal{E}_s| = 0 \end{cases} \tag{4}$$

where R_1 and $R_{0.5}$ are user-specified, and the temporal subgraphs whose ratios of intra-edges are R_1 and $R_{0.5}$ are set to have the InterTC scores of 1 and 0.5 respectively in Equation 4.

Example 4. *Let $R_1 = 0.9$ and $R_{0.5} = 0.4$. Given Figure 3, let $\mathcal{G}_s = I(\{v_1, v_2, v_3, v_4, v_5\})$. Then, $R(\mathcal{G}_s) = \frac{12}{15}$ and $\zeta_{inter}(\mathcal{G}_s) = 0.77$.*

3.4 Intra-Topological Cohesiveness Score ζ_{intra}

The intra-topological cohesiveness score ζ_{intra} (abbr. the IntraTC score) evaluates how densely the vertices in the subgraph are connected, and a denser structure contributes to a larger value of ζ_{intra}. Given a temporal subgraph $\mathcal{G}_s = (V_s, \mathcal{E}_s)$, a basic method to measure its density is to compute $Density(\mathcal{G}_s) = \frac{2|E_s^P|}{|V_s|*(|V_s|-1)}$, where $|E_s^P|$ is the number of edges in the projected graph of $\mathcal{G}_s$. However, as the real-world temporal graphs are often sparse [9], the value of *Density* is always small, and such definition of *Density* is inefficient in distinguishing densely connected subgraphs in real scenes.

Therefore, it is proposed to evaluate the density of a graph with k-core [35]. k-core is a structure utilized in community search algorithms [7, 21], and for a subgraph that is a k-core, the degree of each vertex in it should be larger than $k - 1$. As k-core can evaluate whether vertices are closely engaged in the subgraph, a density function D is proposed based on the k-core structure as follows:

$$D(\mathcal{G}_s) = 1 - \frac{\sum_{v \in V_s} \max\left(k - deg_{G_s^P}(v), 0\right)}{|V_s| * k} = \frac{\sum_{v \in V_s} \min\left(k, deg_{G_s^P}(v)\right)}{|V_s| * k} \tag{5}$$

where $deg_{G_s^P}(v)$ is the degree of v in G_s^P, and $\mathcal{G}_s$ is expected to be a k-core. $D(\mathcal{G}_s)$ is abbreviated as D when there is no ambiguity. In this definition, $\sum_{v \in V_s} \max\left(k - deg_{G_s^P}(v), 0\right)$ is the minimum degrees necessary to convert $\mathcal{G}_s$ into a k-core. Therefore, the density of a temporal subgraph is defined as the ratio of the actual sum of degrees in $\mathcal{G}_s$ to the minimum sum of degrees when $\mathcal{G}_s$ is a k-core. The value of D is in the range of [0, 1], and a larger value of D indicates a denser subgraph. Then, the IntraTC score of $\mathcal{G}_s$ is defined as follows:

$$\zeta_{\text{intra}}(\mathcal{G}_s) = \begin{cases} \frac{1}{1+log\left(\frac{e-1}{D_1-D_{0.5}}*(D_1-\min(D(\mathcal{G}_s),D_1))+1\right)} & |\mathcal{E}_s| \neq 0 \\ 0 & |\mathcal{E}_s| = 0 \end{cases} \tag{6}$$

where D_1 and $D_{0.5}$ are user-specified, and the temporal subgraphs whose densities are D_1 and $D_{0.5}$ are set to have the IntraTC scores of 1 and 0.5 respectively in Equation 6.

Example 5. *Let $k = 3$, $D_1 = 1$, and $D_{0.5} = \frac{2}{3}$. It indicates that the IntraTC score of a 3-core is 1, and that of a temporal subgraph whose vertices all have 2 neighbors is 0.5. For $\mathcal{G}_s = I(\{v_1, v_2, v_3, v_4, v_5\})$ in Figure 3, because $\mathcal{G}_s$ is a 4-core, $\zeta_{intra}(\mathcal{G}_s) = 1$.*

3.5 Computation of $\mathbb{T}$-cohesiveness

The *computeTC*$(\mathcal{G}, \mathcal{G}_s)$ algorithm is proposed to compute the $\mathbb{T}$-cohesiveness of a temporal subgraph $\mathcal{G}_s$ in $\mathcal{G}$. Specifically, $\zeta_t(\mathcal{G}_s)$, $\zeta_{\text{inter}}(\mathcal{G}_s)$, and $\zeta_{\text{intra}}(\mathcal{G}_s)$ are first computed according to Equations 2, 4, and 6, respectively. Next, the $\mathbb{T}$-cohesiveness of $\mathcal{G}_s$ can be computed with $\zeta_t(\mathcal{G}_s) * \zeta_{\text{inter}}(\mathcal{G}_s) * \zeta_{\text{intra}}(\mathcal{G}_s)$. With our carefully designed data structure [27], the time complexities of computing these three scores are $O(|V_s|)$, and that of *computeTC* is also $O(|V_s|)$.

Example 6. *As shown in Figure 3, given $\mathcal{G}_s = I(\{v_1, v_2, v_3, v_4, v_5\})$, it is obtained that $\zeta_t(\mathcal{G}_s) = 0.69$, $\zeta_{inter}(\mathcal{G}_s) = 0.77$, and $\zeta_{intra} = 1$. Then, we have $\mathbb{T}_c(\mathcal{G}_s) = 0.53$.*

3.6 $\mathbb{T}$-cohesiveness for Directed Graphs

The previously proposed $\mathbb{T}$-cohesiveness is for undirected graphs, and it can also be generalized to deal with directed graphs. The main idea is to convert the directed graphs into undirected graphs with some constraints. For example, in a money transfer network, if there is a directed temporal edge between two accounts, the temporal edge is considered to represent a transaction between these two accounts, and the direction of the temporal edge can be omitted. Then, the graph can be converted into an undirected graph. For another example, in a Twitter network, we consider two users to be friends if and only if they follow each other. Then, an undirected graph that represents the friendships can be obtained.

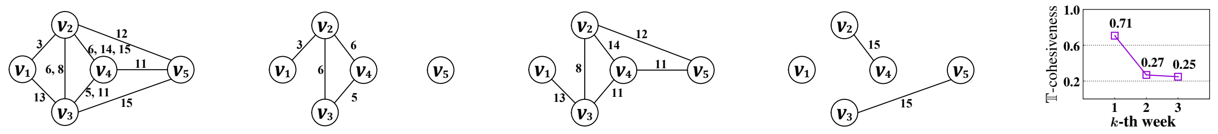
Note that this method is applied in the preprocessing step. After the directed graphs are converted to the undirected graphs, the $\mathbb{T}$-cohesiveness can be easily applied to evaluate their combos.

4 TIME-TOPOLOGY ANALYSIS METHODS

As $\mathbb{T}$-cohesiveness is related to the time and topology dimensions, we conduct time-topology analysis on temporal graphs with it. The idea is partially inspired by the time-frequency analysis technique of signal processing. In this section, two typical time-topology analysis methods are proposed, namely, $\mathbb{T}$-cohesiveness evolution tracking (Subsection 4.1) and combo searching (Subsection 4.2).

4.1 $\mathbb{T}$-cohesiveness Evolution Tracking

The analysis of $\mathbb{T}$-cohesiveness evolution tracking traces the evolution of the $\mathbb{T}$-cohesiveness of a given group of vertices as time goes.



(a) An example of temporal graph. (b) The temporal graph of the first week. (c) The temporal graph of the second week. (d) The temporal graph of the third week. (e) $\mathbb{T}$-cohesiveness of $\{v_2, v_3, v_4\}$ in (a).

Figure 4: An example of performing $\mathbb{T}$-cohesiveness evolution tracking on a temporal graph.

```
Algorithm 1: 𝕋-cohesiveness Evolution Tracking
   Input: $\mathcal{G} = (V, \mathcal{E})$: a temporal graph, $V_s$: a group of vertices, $w$: the width of the time
          window, $st$: the step length of the time window
   Output: $\Gamma$: the list of the 𝕋-cohesiveness of $V_q$ in all the time windows
1  $t_s \leftarrow t_{min}, t_e \leftarrow t_{min} + w$ //the start and end timestamp of the first time window
2  while $t_s < t_{max}$ do
3      $\mathcal{G}_t \leftarrow$ temporal subgraph of $\mathcal{G}$ with all the edges whose timestamps are in $[t_s, t_e]$
4      $\mathcal{G}_s \leftarrow I_{\mathcal{G}_t}(V_s)$
5      $\mathbb{T}_c \leftarrow$ computeTC($\mathcal{G}$, $\mathcal{G}_s$)
6      $\Gamma$.push($\mathbb{T}_c$)
7      $t_s \leftarrow t_s + st, t_e \leftarrow t_s + w$
8  return $\Gamma$
```

It has practical significance and wide applications such as tracking the evolution of organizations and detecting money launderers.

Algorithm 1 shows the process of $\mathbb{T}$-cohesiveness evolution tracking. Specifically, the vertex group is fixed, and the time dimension is changing. The time complexity of Algorithm 1 is $O(|\mathcal{E}| + N_w|V_s|)$, where N_w represents the number of windows. Besides, the space complexity of Algorithm 1 is $O(|V_s| + |V| + |\mathcal{E}| + N_w)$.

Example 7. *The analysis of $\mathbb{T}$-cohesiveness evolution tracking is illustrated with Figure 4. In Figure 4a, each vertex is a person, and the timestamp on an edge is the day the adjacent persons contact. Specifically, $V_s = \{v_2, v_3, v_4\}$ is chosen as the vertex group, and the window width and step length are both set seven days (a week). Besides, we set $T_1 = 3$, $T_{0.5} = 6$, $R_1 = 0.9$, $R_{0.5} = 0.4$, $k = 2$, $D_1 = 1$, and $D_{0.5} = 0.5$. Then, the result of $\mathbb{T}$-cohesiveness evolution tracking is shown in Figure 4e. v_2, v_3, and v_4 are cohesive in the first week (Figure 4b), because these vertices form a 2-core ($\zeta_{intra} = 1$), and there is only one inter-edge ($\zeta_{inter} = 0.71$). Moreover, the timestamps among them are in a short period of time ($\zeta_t = 1$). In the second week (Figure 4c), v_2, v_3, and v_4 are not cohesive, because the timestamps among them are dispersed ($\zeta_t = 0.50$), and there are many inter-edges ($\zeta_{inter} = 0.54$). Besides, in the third week (Figure 4d), v_2, v_3, and v_4 are not cohesive, since the ratio of the intra-edges is still small ($\zeta_{inter} = 0.54$), and the degrees of these three vertices are at most 1 ($\zeta_{intra} = 0.46$).*

4.2 Combo Searching

Given a temporal graph $\mathcal{G}$, a query vertex v, a $\mathbb{T}$-cohesiveness threshold γ, and a number n, the analysis of combo searching aims to find n combos, whose $\mathbb{T}$-cohesiveness values are at least γ. Different from $\mathbb{T}$-cohesiveness evolution tracking, in combo searching, the time dimension is fixed, i.e. all the temporal edges in $\mathcal{G}$ are considered, while the topology dimension (the vertex group) is changing. In this subsection, we first propose an index called Edge-Clustered List to efficiently store the temporal edges (Subsection 4.2.1). Then, the algorithm for combo searching is presented (Subsection 4.2.2).

4.2.1 Edge-Clustered List Index. In combo searching, various vertex groups are generated, and their $\mathbb{T}$-cohesiveness values are evaluated separately to check whether they are combos. As the induced subgraph is considered when the $\mathbb{T}$-cohesiveness of a vertex group is computed, all the temporal edges between two vertices are always processed together. Therefore, an index named Edge-Clustered List (abbreviated as the ECL index), is proposed to store temporal graphs efficiently. An example of an ECL index is shown in Figure 5.

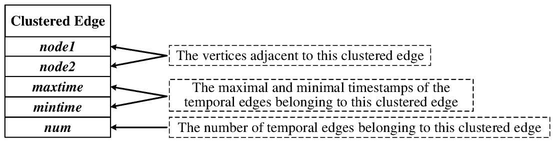
The basic unit of the ECL index is a clustered edge, which contains the statistics of the temporal edges between two vertices. These two vertices are called adjacent to the clustered edge, and the temporal edges are said to belong to the clustered edge. The structure of a clustered edge e^c is shown in Figure 5a, and it consists of five items. First, e^c records its adjacent vertices in *node1* and *node2*. Second, *maxtime* and *mintime* represent the maximal and minimal timestamps of the temporal edges belonging to e^c, respectively. Besides, e^c stores the number of the temporal edges belonging to it in *num*.

For every two vertices with at least one temporal edge between them, a clustered edge is generated. Then, all the clustered edges are ordered in ascending order of their maximal timestamps (the value of *maxtime*). For each vertex u, $ECL(u)$ stores all the clustered edges adjacent to u. With the ECL index, the $\mathbb{T}$-cohesiveness of the temporal subgraphs can be obtained with the clustered edges directly in combo searching, instead of traversing the temporal edges. For instance, when the time span of a temporal subgraph $\mathcal{G}_s$ is computed, the timestamps on all the temporal edges should be collected. However, with the ECL index, only the clustered edges are needed to be traversed, and the time cost is reduced. Moreover, given a temporal graph $\mathcal{G} = (V, \mathcal{E})$, the memory cost of its ECL index is $O(|E^P|)$, which is much lower than that of $\mathcal{G}$ (i.e., $O(|\mathcal{E}|)$). Therefore, when the temporal graph is too large to be stored in memory, we can just store the ECL index to perform combo searching.

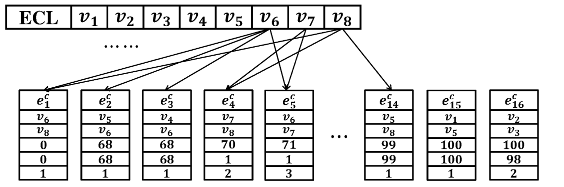
Example 8. *The ECL index built on Figure 3 is shown in Figure 5b, and 16 clustered edges are generated.*

4.2.2 Algorithm for Combo Searching. The *CS* algorithm that performs combo searching is detailed in Algorithm 2. Firstly, Line 1 computes the time span that makes TC score $\zeta_t = \gamma$, and it is noted that every subgraph with a time span longer than *maxspan* should have a value of $\mathbb{T}_c$ smaller than γ, and can never be a combo.

Because each combo is an induced subgraph of a vertex group, in Algorithm 2, different vertex groups are traversed, and it is checked whether their induced subgraphs are combos. The vertex groups are organized with a priority queue Q, and the vertex group with the maximal value of $\mathbb{T}$-cohesiveness is at the top of Q. The vertex group with the query vertex only and the $\mathbb{T}$-cohesiveness of the vertex group (i.e.,$(\{v\}, 0)$) is first added into Q (Line 3). Then, the


Clustered Edge
node1
node2
maxtime
mintime
num
The vertices adjacent to this clustered edge
The maximal and minimal timestamps of the temporal edges belonging to this clustered edge
The number of temporal edges belonging to this clustered edge

(a) The structure of a clustered edge.


ECL
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8
e_1^c v_6 v_8 0 0 1
e_2^c v_5 v_6 68 68 1
e_3^c v_4 v_6 68 68 1
e_4^c v_7 v_8 70 1 2
e_5^c v_6 v_7 71 1 3
e_{14}^c v_5 v_8 99 99 1
e_{15}^c v_1 v_5 100 100 1
e_{16}^c v_2 v_3 100 98 2

$\boldsymbol{ECL}[\boldsymbol{v_1}] = [e_6^c, e_7^c, e_8^c, e_{15}^c], \cdots, \boldsymbol{ECL}[\boldsymbol{v_6}] = [e_1^c, e_2^c, e_3^c, e_5^c], \boldsymbol{ECL}[\boldsymbol{v_7}] = [e_4^c, e_5^c], \boldsymbol{ECL}[\boldsymbol{v_8}] = [e_1^c, e_4^c, e_{14}^c]$

(b) The ECL index of Figure 3.

Figure 5: The Edge-Clustered-List Index.

vertex group with the maximal $\mathbb{T}$-cohesiveness in Q (denoted as g) is retrieved and processed in each iteration (Lines 5–20). Specifically, Line 7 invokes *GetValidNeighbors* to acquire all the valid neighbors of the vertices in g, and the valid neighbors are stored in *validv*, while the invalid neighbors are stored in *invalidv*. Suppose u is a neighbor of a vertex in g, then u is called a valid neighbor of vertices in g iff the time span of $I_{\mathcal{G}}(g \cup \{u\})$ is not longer than *maxspan*. Otherwise, u is an invalid neighbor. The invalid neighbors should never be added into g, because temporal subgraphs containing g and an invalid neighbor of vertices in g cannot be a combo.

For each valid neighbor $u \in validv$, it is added into g to generate a new vertex group *cur* (Line 9), and the $\mathbb{T}$-cohesiveness of $\mathcal{G}_q = I_{\mathcal{G}}(cur)$ is computed (Line 12). If the $\mathbb{T}$-cohesiveness value of $\mathcal{G}_q$ is not smaller than γ, $\mathcal{G}_q$ is a combo, and is added into σ (Line 14). Then, *cur* is added into Q, and can be further extended and processed in the later iterations (Line 20).

The best time complexity of Algorithm 2 is $O(n(|E^P| + |V|))$, where $|E^P|$ is the number of edges in the projected graph of $\mathcal{G}$, while the worst time complexity is $O(2^{|V|}(|E^P| + |V|))$. Moreover, as the space cost of the ECL index is $O(|V| + |E^P|)$, the best space cost of Algorithm 2 is $O(|\mathcal{E}| + |V| + |E^P| + n|V|)$, while the worst space cost is $O(|\mathcal{E}| + |V| + |E^P| + 2^{|V|}|V|)$.

Theorem 1. *The combo searching problem is NP-hard.*

As the combo searching problem is NP-hard, we carefully design our *CS* algorithm to speed up, and *CS* has the following advantages: (1) With the breadth-first-search method, the induced graph of g can be reused at Line 11, and the time cost of constructing $\mathcal{G}_q$ is reduced; (2) By greedily choosing the vertex group with the largest $\mathbb{T}$-cohesiveness in Q to deal with at Line 5, vertex groups with large $\mathbb{T}$-cohesiveness values are more likely to be found earlier; (3) A pruning method named *ComputeMaxTC* is applied to avoid unnecessary computations (proposed in the next section).

Example 9. *The CS algorithm is conducted on Figure 3 (denoted as $\mathcal{G}$) to obtain combos, and the query vertex is v_1. We set T_1 = 4, $T_{0.5}$ = 7, R_1 = 0.9, $R_{0.5}$ = 0.4, k = 3, D_1 = 1, $D_{0.5}$ = $\frac{2}{3}$, and γ = 0.5. Please note that in the following examples, the parameters are also set like this. First, as $T_{0.5}$ = 7 and γ = 0.5, maxspan = 7 days is obtained at Line 1. Line 3 initializes the priority queue Q with the query vertex,*

Algorithm 2: Combo Searching (*CS*)

```
    Input: $\mathcal{G} = (V, \mathcal{E})$: a temporal graph, $v$: the query vertex, $\gamma$: the threshold of
           $\mathbb{T}$-cohesiveness, $n$: the number of combos to obtain, $ECL$: the ECL index of $\mathcal{G}$
    Output: $\sigma$: $n$ combos
1   $maxspan \leftarrow \frac{(e^{\frac{1}{\gamma}-1}-1)*(T_{0.5}-T_1)}{e-1} + T_1$
2   $Q$ ← a priority queue of vertex groups, vertex groups are sorted by their $\mathbb{T}$-cohesiveness
3   $Q.push((\{v\}, 0))$ //the first vertex group
4   while $|\sigma| < n$ and $|Q| > 0$ do
5       $(g, \mathbb{T}_c(I(g))) \leftarrow Q.Top()$
6       $Q.Pop()$
7       $validv, invalidv \leftarrow GetValidNeighbors(\mathcal{G}, g, maxspan, ECL)$
8       foreach vertex $u \in validv$ do
9           $cur \leftarrow g \cup \{u\}$
10          if cur is never visited before then
11              $\mathcal{G}_q \leftarrow I_{\mathcal{G}}(cur)$
12              $\mathbb{T}_c \leftarrow computeTC(\mathcal{G}, \mathcal{G}_q)$
13              if $\mathbb{T}_c \geq \gamma$ then
14                  $\sigma.Push(\mathcal{G}_q)$
15                  if $|\sigma| \geq n$ then
16                      return $\sigma$
17              // Compute the upper bound of $\mathbb{T}$-cohesiveness. Lines 18–19 are omitted
                  when optimization is not applied
18              $\widehat{\mathbb{T}}_c \leftarrow ComputeMaxTC(\mathcal{G}, cur, ECL, \gamma, maxspan)$
19              if $\widehat{\mathbb{T}}_c \geq \gamma$ then
20                  $Q.Push((cur, \mathbb{T}_c))$
21  return $\sigma$
```

i.e., $Q = [(\{v_1\}, 0)]$. Then, Lines 4–20 traverse all the possible vertex groups and compute their $\mathbb{T}$-cohesiveness. In the first iteration, $g = \{v_1\}$, and the valid neighbors of g are validv = $\{v_2, v_3, v_4, v_5\}$. Each vertex in validv is added into g separately, and new vertex groups can be generated. Specifically, four new vertex groups $\{v_1, v_2\}$, $\{v_1, v_3\}$, $\{v_1, v_4\}$, $\{v_1, v_5\}$ are generated, and their $\mathbb{T}$-cohesiveness values are calculated at Line 12. Because their $\mathbb{T}$-cohesiveness values are all 0.17 < γ, their induced subgraphs are not stored into σ. However, these vertex groups may form combos after some more vertices are added into them. For instance, the $\mathbb{T}$-cohesiveness value of $\{v_1, v_2, v_3, v_4, v_5\}$ is larger than γ. Therefore, these four new vertex groups are pushed into Q (Line 20), and processed in the next iterations.

5 OPTIMIZATION

In this section, an optimization method is proposed to prune the vertex groups that can never become cohesive enough no matter which vertices are added into them. Before the optimization method is presented, the concept of a supergroup is first proposed.

Definition 6 (supergroup). *Given two vertex groups g and g', g' is called the supergroup of g if and only if $g \subset g'$.*

The relationship of supergroup is transitive. In Algorithm 2, when vertex group g is processed, the new vertex groups *cur* generated at Line 9 are all supergroups of g. Moreover, after *cur* is added into Q, and when it is processed in the later iterations, the new vertex groups generated are also supergroups of g.

Also in Algorithm 2, every newly obtained vertex group *cur* is inserted into the priority queue Q and processed in the later iterations (Line 20). However, it is noted that there are some vertex groups whose supergroups cannot have $\mathbb{T}$-cohesiveness not smaller than γ. These vertex groups are called *invalid vertex groups.* It is obvious that the invalid vertex groups should not be pushed into Q.

Example 10. *Let cur = $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ be a vertex group in Figure 3, and $\overline{cur}$ is a supergroup of cur. We have maxspan = 7 days.*

The time span of $I(cur)$ is 32 days, so the time span of $I(\overline{cur})$ is no shorter than 32 days, and the TC score of $I(\overline{cur})$ is smaller than γ. Because $\overline{cur}$ can be any supergroup of cur, the $\mathbb{T}$-cohesiveness values of all the supergroups of cur are smaller than γ. It indicates that cur is an invalid vertex group and should not be pushed into Q.

Therefore, in this section, for each vertex group *cur* generated, the upper bound of the $\mathbb{T}$-cohesiveness of *cur's* supergroups is computed, and the upper bound is denoted as $\widehat{\mathbb{T}}_c$. If $\widehat{\mathbb{T}}_c$ is smaller than γ, then *cur* is an invalid vertex group, and should be dropped.

As defined in Equation 1, $\mathbb{T}$-cohesiveness consists of three terms: ζ_t, ζ_{inter}, and ζ_{intra}. Therefore, for each vertex group *cur*, the upper bounds of ζ_t, ζ_{inter}, and ζ_{intra} of its supergroups, denoted as $\widehat{\zeta_t}$, $\widehat{\zeta}_{\text{inter}}$, and $\widehat{\zeta}_{\text{intra}}$ respectively, are first computed (Subsections 5.1–5.3), and then $\widehat{\mathbb{T}}_c$ of *cur's* supergroups is obtained (Subsection 5.4).

5.1 Computation of $\widehat{\zeta_t}$

Given a vertex group *cur* and a supergroup $\overline{cur}$ of *cur*, because the temporal edges in $I(cur)$ is a subset of the temporal edges in $I(\overline{cur})$, we have the following theorem.

Theorem 2. *The TC score of a vertex group is no smaller than those of its supergroups.*

It means that when the given vertex group *cur* is extended to generate one of its supergroups $\overline{cur}$, the TC score of $I(\overline{cur})$ is no larger than that of *cur*. Therefore, we have $\widehat{\zeta_t} = \zeta_t(I(cur))$.

Example 11. *In Figure 3, suppose cur* $= \{v_1, v_2, v_3, v_4\}$*, because the time span of* $I(cur)$ *is five days,* $\widehat{\zeta_t} = \zeta_t(I(cur)) = 0.69$.

5.2 Computation of $\widehat{\zeta}_{\text{inter}}$

As shown in Equation 4, the upper bound of ζ_{inter} is determined by that of R (denoted as $\widehat{R}$). Therefore, the problem of computing $\widehat{\zeta}_{\text{inter}}$ becomes to compute the value of $\widehat{R}$. In this subsection, we first propose some concepts, and then analyze the value of $\widehat{R}$.

Definition 7 (Sound Clustered Edge). *Given temporal graph $\mathcal{G}$, vertex group cur, and vertex $u \notin cur$, a clustered edge e^c adjacent to u is sound w.r.t. cur iff it satisfies (1)* ***temporal constraint:*** $\max(maxt, e^c.maxtime) - \min(mint, e^c.mintime) \le maxspan$*, where maxt and mint are the maximal and minimal timestamps of $I(cur)$ respectively; (2)* ***vertex constraint:*** *e^c.node1 and e^c.node2 are not invalid neighbors of vertices in cur. Otherwise, e^c is unsound.*

For a vertex $v_i \notin cur$, $N^{c(v_i)}_{sound}$ and $N^{c(v_i)}_{unsound}$ represent the number of sound and unsound clustered edges (w.r.t. *cur*) adjacent to v_i, respectively. Note that for each supergroup $\overline{cur}$ of *cur*, if $I(\overline{cur})$ contains a temporal edge belonging to an unsound clustered edge, we have $\zeta_t(I(\overline{cur})) < \gamma$, and then $\mathbb{T}_c(I(\overline{cur})) < \gamma$. Such supergroups cannot make *cur* valid and are negligible in estimating the $\widehat{\mathbb{T}}_c$ value of *cur*. Therefore, these supergroups are omitted, and the supergroups mentioned in the rest of this paper are those whose induced subgraphs do not have unsound clustered edges w.r.t. *cur*.

Definition 8 (Sound Edges). *Given a temporal graph $\mathcal{G}$, a vertex group cur, and a vertex $u \notin cur$, a temporal edge e adjacent to u is sound w.r.t. cur if and only if the clustered edge to which e belongs is sound. Otherwise, e is an unsound edge.*

Algorithm 3: ComputeMaxTC

```
Input: $\mathcal{G} = (V, \mathcal{E})$: a temporal graph, cur: a vertex group, ECL: ECL index on $\mathcal{G}$, $\gamma$: the
       threshold of $\mathbb{T}$-cohesiveness, maxspan: the maximal time span a combo can have
Output: $\widehat{\mathbb{T}}_c$: the upper bound of the $\mathbb{T}$-cohesiveness of the supergroups of cur
1 $\widehat{\zeta_t}$ = ComputeZetaT($\mathcal{G}$, cur) //Compute $\widehat{\zeta_t}$, quit if $\widehat{\zeta_t} < \gamma$
2 validv, invalidv ← GetValidNeighbors($\mathcal{G}$, cur, maxspan, ECL)
3 $\widehat{\zeta}_{\text{inter}}$ = ComputeZetaInter($\mathcal{G}$, cur, ECL, maxspan, invalidv) //quit if $\widehat{\zeta}_{\text{inter}} < \gamma$
4 $\widehat{\zeta}_{\text{intra}}$ = ComputeZetaIntra($\mathcal{G}$, cur, ECL, maxspan, invalidv) //quit if $\widehat{\zeta}_{\text{intra}} < \gamma$
5 $\widehat{\mathbb{T}}_c = \widehat{\zeta_t} * \widehat{\zeta}_{\text{inter}} * \widehat{\zeta}_{\text{intra}}$ // Compute the upper bound of the $\mathbb{T}$-cohesiveness
6 return $\widehat{\mathbb{T}}_c$
```

For a vertex $v_i \notin cur$, $N^{(v_i)}_{sound}$ represents the number of sound edges w.r.t. *cur* adjacent to v_i, and $N^{(v_i)}_{unsound}$ represents the number of unsound edges adjacent to v_i. The unsound edges w.r.t. *cur* should never appear in the induced subgraph of any supergroup of *cur*.

Example 12. *In Figure 3, suppose cur* $= \{v_1, v_2, v_3, v_4\}$ *and maxspan = 7 days. For vertex $v_5 \notin cur$, the clustered edge between v_5 and v_6 is unsound, because v_6 is an invalid neighbor of vertices in cur. Then, $(v_5, v_6, 68)$ is an unsound edge. Besides, the other clustered edges adjacent to v_5 are sound, therefore, temporal edges $(v_1, v_5, 100)$, $(v_2, v_5, 97)$, $(v_3, v_5, 99)$, $(v_4, v_5, 99)$, and $(v_5, v_8, 99)$ are sound edges w.r.t. cur. For vertex v_7, the clustered edges adjacent to v_7 are both unsound clustered edges, because they violate the temporal constraint. Therefore, the five temporal edges adjacent to v_7 are all unsound edges.*

Theorem 3. *Given $\mathcal{G} = (V, \mathcal{E})$ and a vertex group cur $\subset V$, suppose $v_{n_1}, \cdots, v_{n_s}$ are all the vertices in $V-$ cur, and satisfy $\frac{N^{(v_{n_1})}_{sound}}{N^{(v_{n_1})}_{unsound}} \ge \cdots \ge \frac{N^{(v_{n_s})}_{sound}}{N^{(v_{n_s})}_{unsound}}$. Then, if $\frac{N^{(v_{n_1})}_{sound}}{N^{(v_{n_1})}_{unsound}} \ge \frac{|\mathcal{E}_{cur}|}{|\mathcal{E}^l_{cur}|}$, $\widehat{R} = \frac{|\mathcal{E}_{cur}| + \sum_{1\le j\le i} N^{(v_{n_j})}_{sound}}{|\mathcal{E}^l_{cur}| + \sum_{1\le j\le i} N^{(v_{n_j})}_{unsound}}$, where v_{n_i} is the first vertex in $v_{n_1}, \cdots, v_{n_s}$ satisfying $\frac{|\mathcal{E}_{cur}| + \sum_{1\le j\le i} N^{(v_{n_j})}_{sound}}{|\mathcal{E}^l_{cur}| + \sum_{1\le j\le i} N^{(v_{n_j})}_{unsound}} \ge \frac{N^{(v_{n_{i+1}})}_{sound}}{N^{(v_{n_{i+1}})}_{unsound}}$. Otherwise, $\widehat{R} = \max\left(\frac{|\mathcal{E}_{cur}| + N^{(v_{n_1})}_{sound}}{|\mathcal{E}^l_{cur}| + N^{(v_{n_1})}_{unsound}}, \cdots, \frac{|\mathcal{E}_{cur}| + N^{(v_{n_s})}_{sound}}{|\mathcal{E}^l_{cur}| + N^{(v_{n_s})}_{unsound}}\right)$.*

According to Theorem 3, $\widehat{\zeta}_{\text{inter}} = \frac{1}{1+log(\frac{e-1}{R_1-R_{0.5}} * (R_1 - \min(\widehat{R}, R_1)) + 1)}$.

Example 13. *In Figure 3 (i.e. $\mathcal{G} = (V, \mathcal{E})$), let cur* $= \{v_1, v_2, v_3, v_4\}$*. Then, $|\mathcal{E}_{cur}| = 8$, $|\mathcal{E}^l_{cur}| = 13$, and maxspan $= 7$. For the vertices in $V-cur$, $\frac{N^{(v_5)}_{sound}}{N^{(v_5)}_{unsound}} = \frac{5}{1}$, $\frac{N^{(v_6)}_{sound}}{N^{(v_6)}_{unsound}} = \frac{0}{6}$, $\frac{N^{(v_7)}_{sound}}{N^{(v_7)}_{unsound}} = \frac{0}{5}$, and $\frac{N^{(v_8)}_{sound}}{N^{(v_8)}_{unsound}} = \frac{1}{3}$. Therefore, $\widehat{R} = \frac{8+5}{13+1} = 0.93$, and $\widehat{\zeta}_{inter} = 1$.*

Moreover, let cur $= \{v_1, v_2, v_3, v_4, v_5\}$*, we have $|\mathcal{E}_{cur}| = 12$, $|\mathcal{E}^l_{cur}| = 15$, and $V-cur = \{v_6, v_7, v_8\}$. Then, $\frac{|\mathcal{E}_{cur}| + N^{(v_{n_8})}_{sound}}{|\mathcal{E}^l_{cur}| + N^{(v_{n_8})}_{unsound}}$ is the maximal, $\widehat{R} = \frac{12+1}{15+3} = 0.72$, and $\widehat{\zeta}_{inter} = 0.68$.*

5.3 Computation of $\widehat{\zeta}_{\text{intra}}$

In Equation 6, $\widehat{\zeta}_{\text{intra}}$ is determined by the upper bound of D (denoted as $\widehat{D}$), and $\widehat{D}$ can be obtained with the following theorem.

Theorem 4. *Given $\mathcal{G} = (V, \mathcal{E})$, a vertex group cur $\subset V$, and core number k, suppose $v_{n_1}, \cdots, v_{n_s}$ are all the vertices in $V-cur$, and satisfy*

Table 2: Statistics of Datasets

Dataset	$\|V\|$	$\|E\|$	$\|\mathcal{E}\|$	T	Dataset	$\|V\|$	$\|E\|$	$\|\mathcal{E}\|$	T
Col-1d	1,899	102	314	1d	Cont-7d	10,972	7,170	110,134	7d
Col-7d	1,899	326	1,838	7d	Cont-30d	10,972	18,164	322,268	30d
Col-30d	1,899	1,069	6,646	30d	Dblp-4y	364,605	1,032,437	2,700,430	4y
Email-1d	986	592	1,904	1d	Syn_{1000}	1,000	1,270	1,295	1d
Email-7d	986	1,258	6,488	7d	Syn_{10000}	10,000	9,345	10,461	1d
Email-30d	986	1,938	13,942	30d	Syn_{100000}	100,000	627,798	838,121	1d
Cont-1d	10,972	3,003	35,994	1d	Syn_{1M}	1,000,000	7,789,875	11,442,996	1d

Table 3: Configurations of Parameters

CF. ID	$(T_1, T_{0.5})$	$(R_1, R_{0.5})$	$(k, D_1, D_{0.5})$	CF. ID	$(T_1, T_{0.5})$	$(R_1, R_{0.5})$	$(k, D_1, D_{0.5})$
1 (*)	(0.5d, 1d)	(0.9, 0.4)	(2, 1, 0.5)	15 (*)	(14d, 28d)	(0.9, 0.3)	(2, 1, 0.5)
2	(**0.4d**, 1d)	(0.9, 0.4)	(2, 1, 0.5)	16	(**13d**, 28d)	(0.9, 0.3)	(2, 1, 0.5)
3	(0.5d, **0.9d**)	(0.9, 0.4)	(2, 1, 0.5)	17	(14d, **27d**)	(0.9, 0.3)	(2, 1, 0.5)
4	(0.5d, 1d)	(**0.95**, 0.4)	(2, 1, 0.5)	18	(14d, 28d)	(**0.95**, 0.3)	(2, 1, 0.5)
5	(0.5d, 1d)	(0.9, **0.5**)	(2, 1, 0.5)	19	(14d, 28d)	(0.9, **0.35**)	(2, 1, 0.5)
6	(0.5d, 1d)	(0.9, 0.4)	(**3**, 1, **2/3**)	20	(14d, 28d)	(0.9, 0.3)	(**3**, 1, **2/3**)
7	(0.5d, 1d)	(0.9, 0.4)	(2, **0.95**, 0.5)	21	(14d, 28d)	(0.9, 0.3)	(2, **0.95**, 0.5)
8 (*)	(4d, 7d)	(0.9, 0.4)	(2, 1, 0.5)	22 (*)	(0y, 2y)	(0.9, 0.3)	(2, 1, 0.5)
9	(**3.5d**, 7d)	(0.9, 0.4)	(2, 1, 0.5)	23	(**1y**, 2y)	(0.9, 0.3)	(2, 1, 0.5)
10	(4d, **6d**)	(0.9, 0.4)	(2, 1, 0.5)	24	(0y, **1y**)	(0.9, 0.3)	(2, 1, 0.5)
11	(4d, 7d)	(**0.95**, 0.4)	(2, 1, 0.5)	25	(0y, 2y)	(**0.95**, 0.3)	(2, 1, 0.5)
12	(4d, 7d)	(0.9, **0.5**)	(2, 1, 0.5)	26	(0y, 2y)	(0.9, **0.35**)	(2, 1, 0.5)
13	(4d, 7d)	(0.9, 0.4)	(**3**, 1, **2/3**)	27	(0y, 2y)	(0.9, 0.3)	(**3**, 1, **2/3**)
14	(4d, 7d)	(0.9, 0.4)	(2, **0.95**, 0.5)	28	(0y, 2y)	(0.9, 0.3)	(2, **0.95**, 0.5)

$\frac{\min(k, N_{sound}^{c(v_{n_1})})}{k} \geq \cdots \geq \frac{\min(k, N_{sound}^{c(v_{n_s})})}{k}$. *Let* $C = \sum_{v \in cur} \min(k, N_{sound}^{c(v)})$. *If* $\frac{\min(k, N_{sound}^{c(v_{n_1})})}{k} \geq \frac{C}{|cur|*k}$, $\widehat{D} = \frac{C + \sum_{1 \leq j \leq i} \min(k, N_{sound}^{c(v_{n_j})})}{|cur|*k+i*k}$, *where* v_{n_i} *is the first vertex in* $v_{n_1}, \cdots, v_{n_s}$ *satisfying* $\frac{C + \sum_{1 \leq j \leq i} \min(k, N_{sound}^{c(v_{n_j})})}{|cur|*k+i*k} \geq \frac{\min(k, N_{sound}^{c(v_{n_{i+1}})})}{k}$. *Otherwise,* $\widehat{D} = \max(\frac{C+\min(k, N_{sound}^{c(v_{n_1})})}{|cur|*k+k}, \cdots, \frac{C+\min(k, N_{sound}^{c(v_{n_s})})}{|cur|*k+k})$.

According to Theorem 4, $\widehat{\zeta}_{\text{intra}} = \frac{1}{1+log(\frac{e-1}{D_1-D_{0.5}}*(D_1-\min(\widehat{D}, D_1))+1)}$.

Example 14. *In Figure 3, let* $cur = \{v_1, v_2, v_3, v_4\}$. *Then,* $C = 12$, $\frac{C}{|cur|*k} = 1$. *For vertices in* $V - cur$, $\frac{\min(k, N_{sound}^{c(v_5)})}{k} = \frac{3}{3}$, $\frac{\min(k, N_{sound}^{c(v_6)})}{k} = \frac{\min(k, N_{sound}^{c(v_7)})}{k} = \frac{0}{3}$, *and* $\frac{\min(k, N_{sound}^{c(v_8)})}{k} = \frac{1}{3}$. *Then,* $\widehat{D} = 1$ *and* $\widehat{\zeta}_{intra} = 1$. *Moreover, suppose that* $cur = \{v_1, v_2, v_3, v_4, v_5\}$, *we have* $C = 15$ *and* $\frac{C}{|cur|*k} = 1$. *Then,* $V - cur = \{v_6, v_7, v_8\}$. *Since* $\frac{\min(k, N_{sound}^{c(v_8)})}{k}$ *is the maximal,* $\widehat{D} = \frac{15+1}{5*3+3} = 0.89$, *and* $\widehat{\zeta}_{intra} = 0.69$.

Lemma 1. *Vertex groups with* $\min(\widehat{\zeta}_t, \widehat{\zeta}_{inter}, \widehat{\zeta}_{intra}) < \gamma$ *are invalid.*

5.4 $\mathbb{T}$-cohesiveness Upper Bound

With $\widehat{\zeta}_t$, $\widehat{\zeta}_{\text{inter}}$, and $\widehat{\zeta}_{\text{intra}}$ analyzed above, an algorithm named *ComputeMaxTC* (Algorithm 3) is proposed to compute the $\mathbb{T}$-cohesiveness upper bound of the supergroups of a given vertex group.

The time complexities of *ComputeZetaT*, *ComputeZetaInter*, and *ComputeZetaIntra* are $O(|E^P|)$, $O(|E^P| + |V|)$, and $O(|E^P| + |V|)$, respectively. Then, that of Algorithm 3 is $O(|E^P| + |V|)$.

Example 15. *Given Figure 3, suppose that* $cur = \{v_1, v_2, v_3, v_4, v_5\}$ *is obtained at Line 9 of Algorithm 2. When ComputeMaxTC is invoked at Line 18, we obtain that* $\widehat{\zeta}_t = 0.69$, $\widehat{\zeta}_{inter} = 0.68$ *and* $\widehat{\zeta}_{intra} = 0.69$, $\widehat{\mathbb{T}}_c = 0.32 < \gamma = 0.5$. *Therefore, the supergroups of cur cannot form a combo, and cur is not inserted into Q.*

6 EXPERIMENTS

In this section, datasets and parameter configurations used in the experiments are first introduced in Section 6.1. Then, the results of the performance evaluation, memory cost evaluation, ablation study, scalability evaluation, parameter sensitivity, and case study are reported and analyzed in Sections 6.2–6.7.

6.1 Experimental Setting

Ten real-world datasets and four synthetic datasets are used in our experiments. The real-world datasets are extracted from ***CollegeMsg*** [29], ***Email*** [30], ***Contact*** [34], and ***DBLP*** networks. ***CollegeMsg*** is a messaging network at a university, and ***Email*** is an email network from a European research institute. ***Contact*** is a human contact network, where the temporal edges represent the proximity of persons. ***DBLP*** records the coauthorships among researchers. There is an edge between two researchers if they coauthor an article, and the timestamp on this edge is the publishing year of this article. Directed temporal graphs are converted into undirected versions by considering the directed edges as undirected relationships (e.g., a directed edge in ***Email*** represents a communication between two persons) as stated in Section 3.6.

Taking Col-1d as an example, it is generated in the following steps: (1) Put all the vertices of ***CollegeMsg*** into Col-1d; (2) Set a start timestamp t_s and add all the temporal edges with existing timestamps in the range of $[t_s, t_s + 86400$ seconds$]$ into Col-1d. The other nine real-world datasets are generated in the similar way.

The synthetic datasets are generated in two steps. Specifically, normal graphs without timestamps are first generated, and then the randomly generated timestamps in the range of [1, 86400] are assigned to the edges. The unit of timestamp is second.

Table 2 shows the statistics of the datasets, where $|V|$, $|E|$, and $|\mathcal{E}|$ are the numbers of vertices, normal edges, and temporal edges in the temporal graph, respectively. T is the time span of the temporal graph. 'd' and 'y' are the short forms of 'day' and 'year', respectively. Besides, for Dblp-4y, the unit of timestamp is one year, while for each of the other datasets, the unit of timestamp is one second.

Moreover, the used parameter configurations (abbreviated as CF.) are shown in Table 3. Every seven configurations compose a group, and the first one in a group is the base and marked as "(*)". For example, CF. 1–CF. 7 form a group, and CF. 1 is the base. Specifically, CF. 2–CF. 7 are obtained by changing the values of T_1, $T_{0.5}$, R_1, $R_{0.5}$, k, and D_1 of CF. 1, respectively. The changed terms are **bold**. Note that $D_{0.5}$ is set based on k to make a temporal subgraph, whose vertices all have k-1 neighbors, have an IntraTC score of 0.5.

Our experiments are carried out on a server with Intel Xeon E5-2650 2.0GHz CPU and 256GB RAM.

6.2 Performance Evaluation

The performances of the proposed two time-topology analysis methods are evaluated on real-world datasets. The parameter configurations used on the datasets with the time spans of 7d and 30d are CF. 8 and CF. 15, respectively, and that on Dblp-4y is CF. 22. The experimental results on Col-1d, Email-1d, and Cont-1d are omitted due to the length limitation.

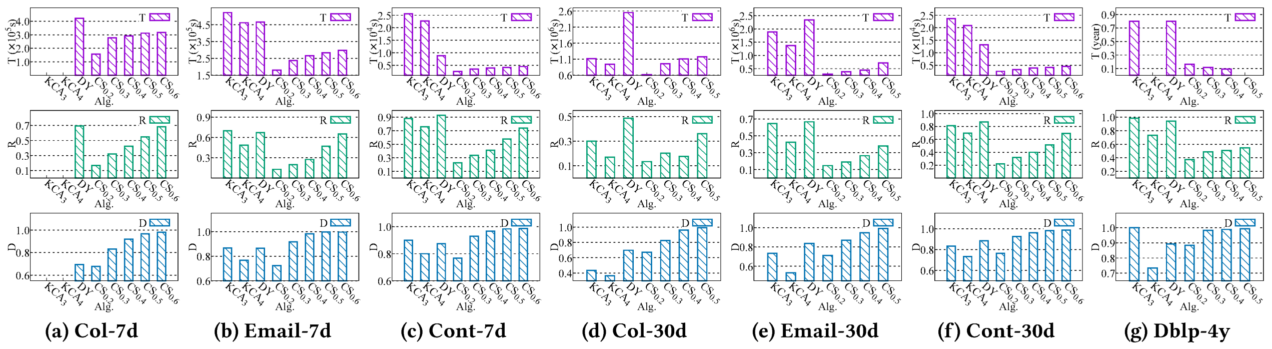


Figure 6: The results of the performance experiments. T, R, and D represent the average time spans, ratios of the intra-edges, and densities of the obtained combos. Their values are reported with purple, green, and blue histograms, respectively.

In the experiments, we perform combo searching on each dataset, and set γ to be 0.2, 0.3, 0.4, 0.5, or 0.6. For each γ value and a dataset, there are 30 query vertices, and 30 combos are obtained for each query vertex with the CS algorithm (Algorithm 2). For each obtained combo $\mathcal{G}_s$, its time span, ratio of intra-edges, and density (denoted as T, R, and D respectively) are evaluated. Note that a smaller value of T, and larger values of R and D indicate a more cohesive subgraph. Finally, given a dataset, the average values of Ts, Rs, and Ds of the obtained combos are reported. For simplicity, these averaged values are also called the T, R, and D of CS on the dataset.

Three baselines are used in the experiments. The first two are the k-core community search algorithms [7] (abbr. KCAs), and denoted as KCA_3 and KCA_4 respectively. Given a temporal graph $\mathcal{G}$ and a query vertex u, KCA_k finds the maximal k-core community that contains u. The third baseline is Dynamo [46], which is proposed for dynamic community detection based on the widely used Louvain algorithm [2]. Since combo searching finds combos of the temporal graph (rather than combos of any snapshot), Dynamo degrades into Louvain in the process, and uses the whole temporal graph to find combos. For these three baselines, given a temporal graph and a query vertex u, the community to which u belongs is considered the combo of u, and its T, R and D are recorded. The experimental results are shown in Figure 6. Note that DY represents Dynamo, and CS_x represents our CS algorithm with $\gamma = x$. The results of $CS_{0.6}$ are not reported in Figure 6d, 6e, and 6g, because 30 combos cannot be obtained in an hour for most query vertices when $\gamma = 0.6$.

Performance of $\mathbb{T}$-cohesiveness evolution tracking. We confirm the good performance of $\mathbb{T}$-cohesiveness evolution tracking by showing $\mathbb{T}$-cohesiveness can correctly depict the cohesiveness of temporal graphs. As shown in Figure 6, with the increase of γ, the T, R, and D values of CS always increase, which indicates a temporal subgraph with a larger $\mathbb{T}$-cohesiveness always has a larger ratio of intra-edges, and the vertices in it are more densely connected, while its time span is a bit longer. The reason for the increase of T may be that given a temporal subgraph $\mathcal{G}_s$, larger values of R and D can be obtained by adding into $\mathcal{G}_s$ some vertices that have many connections with the vertices in $\mathcal{G}_s$, and the new temporal edges may make the time span of the new temporal graph longer. Although the T values become larger with the increase of γ, they are still shorter than the corresponding T_1s (i.e., their TC scores are 1). It indicates that the obtained subgraphs are still temporally cohesive and more topologically cohesive (having large values of R and D). The result on Dblp-4y is a special case, and the value of T decreases when γ increases. The reason may be that CF. 22 is applied on Dblp-4y, and T_1 = 0y. It indicates when T increases, the TC score will decrease to be much smaller than 1, and $\mathbb{T}$-cohesiveness of the subgraph is smaller than γ. Then, on Dblp-4y, the obtained combos have smaller Ts when γ increases, and are more cohesive in both the time and topology dimensions. Therefore, a temporal subgraph with a larger $\mathbb{T}$-cohesiveness is always more cohesive.

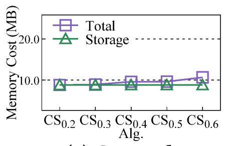
Hence, the above analysis shows that $\mathbb{T}$-cohesiveness can well evaluate the cohesiveness of temporal subgraphs. Besides, because subgraphs with quite large time spans are considered cohesive by KCAs and Dynamo (e.g., Email-7d, Email-30d, and Cont-30d), it is demonstrated that KCAs and Dynamo are poor at evaluating the cohesiveness of temporal subgraphs, and our $\mathbb{T}$-cohesiveness evolution tracking method is superior to the baselines.

Performance of combo searching. The experimental results also suggest that CS has better performance than the baseline methods when the three terms T, R, and D are considered simultaneously. Note that the baselines are compared with $CS_{0.5}$ on Col-30d, Email-30d, and Dblp-4y, and compared with $CS_{0.6}$ on the other datasets. The combos found by Dynamo always have larger Ts and smaller Ds than those obtained by CS (e.g., on Col-7d). In terms of the R values, CS is similar as Dynamo. Because Dynamo focuses on maximizing the R value when finding combos, it sometimes has larger R than CS, e.g., on Col-30d and Email-30d. However, on these datasets, the combos obtained by CS have much smaller Ts and larger Ds than Dynamo, and it demonstrates that the combos found by CS are significantly more temporally cohesive and intra-topologically cohesive. Therefore, CS has better performance than Dynamo.

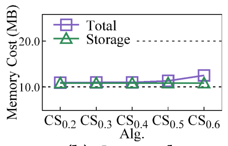
Meanwhile, CS also has better performance than the KCAs. Specifically, for KCA_3, the experimental results show that the combos obtained by CS often have smaller T values and larger D values than those obtained by KCA_3, and are more cohesive. A special case is that on Col-7d, the T value of KCA_3 is 0, and smaller than that of CS. The reason is that there are not any 3-cores that contain the query vertices on Col-7d, and the vertices themselves are returned as the combos. Then, the T, R, and D values of the combos are all 0. In terms of R, KCA_3 sometimes has larger R than CS, e.g., on Email-7d and Cont-7d. The reason is that for some query vertices, the connected components to which the vertices belong are obtained

Table 4: Memory cost of $\mathbb{T}$-cohesiveness evolution tracking.

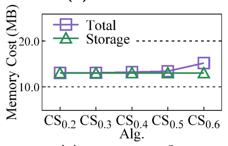
Group Size	Memory Cost on Different Datasets (MB)								
	Col-1d	Col-7d	Col-30d	Email-1d	Email-7d	Email-30d	Cont-1d	Cont-7d	Cont-30d
10	3.5	3.5	3.9	4.5	5.1	5.5	9.8	11.0	15.0
100	3.5	3.6	3.9	4.5	5.5	5.7	9.9	11.3	15.5
1,000	3.6	3.6	4.2	4.7	5.5	5.8	10.1	11.4	16.1



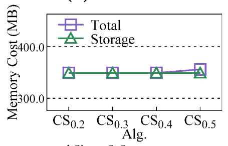
(a) Cont-1d



(b) Cont-7d



(c) Cont-30d



(d) Dblp-4y

Figure 7: Memory cost of combo searching.

as the combos, and the components always have a large ratio of intra-edges. However, a vertex itself or a connected component is usually not a meaningful subgraph, and the combos obtained by *CS* are more instructive and significant. The results of KCA_4 are similar to those of KCA_3. The main difference is that the combos obtained by KCA_4 (i.e. 4-cores) always have smaller T, R, and D values than those obtained by KCA_3 (i.e. 3-cores), because the 4-cores always contain fewer vertices and edges than the 3-cores.

Besides the performance on the evaluation metrics of T, R, and D, the baselines have more significant drawbacks: (1) For a query vertex, the baselines can find only one combo, while our method allows the users to specify the number of combos to be found; (2) The baselines neglect the temporal information in temporal graphs, and a subgraph with a quite large time span may still be considered cohesive; (3) The baselines cannot adapt to the varied conditions, while $\mathbb{T}$-cohesiveness allows users to specify parameters such as T_1 and γ to find combos suitable for different conditions.

6.3 Memory Cost Evaluation

The memory costs (*mc*) of $\mathbb{T}$-cohesiveness evolution tracking and combo searching are also evaluated. The configurations applied are the same as those used in Section 6.2. Moreover, on each dataset, 30 vertices are queried, and their average *mc* is reported.

For $\mathbb{T}$-cohesiveness evolution tracking, both the width and the step length of the time window are set to one day. As shown in Table 4, *mc* increases slightly with the increase of the group size. Besides, with the increase of the temporal edge number (e.g., Cont-1d $\rightarrow$ Cont-7d $\rightarrow$ Cont-30d), *mc* becomes larger. The results are consistent with the space complexity presented in Section 4.1.

For combo searching, the results are shown in Figure 7. Only the results on Cont-1d, Cont-7d, Cont-30d, and Dblp-4y are reported, because the results on the other datasets are similar. The purple lines represent the total *mc*, and the green lines represent the *mc* of the datasets and indexes. Then, the difference between the purple line and the green line is the runtime *mc*. The results show that the runtime *mc* becomes larger with the increase of γ. It is because with a larger γ, combos are more difficult to be found, and more

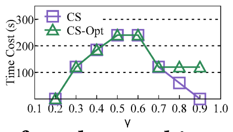


Figure 8: Time cost of combo searching using *CS* and *CS-Opt*.

vertex groups are generated and stored in Q, which contributes to a larger *mc*. However, note that the *mc*s of combo searching are always reasonable on all the tested datasets. Specifically, the total *mc*s are smaller than 20 MB on the first three datasets, and that of processing Dblp-4y is also smaller than 360 MB.

6.4 Ablation Study

In this subsection, we illustrate the efficiency of our optimization method presented in Section 5. In detail, we denote Algorithm 2 by *CS*, and denote the algorithm without the optimization method (i.e., removing Lines 18–19 in Algorithm 2) by *CS-Opt*. Experiments are conducted on Email-30d, and CF. 20 is used. Given $\gamma \in \{0.2, 0.3, \cdots, 0.9\}$, 30 query vertices are queried, and 30 combos are required for each query vertex. The average time cost of obtaining 30 combos for a query vertex is reported. For each query vertex, if 30 combos cannot be found in 30 min, the algorithm is stopped, and the time cost of this vertex is recorded as 30 min.

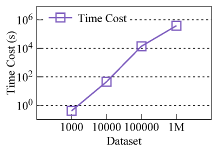
As shown in Figure 8, the experimental results suggest that our optimization method can accelerate combo searching significantly, and the optimization effect becomes better with the increase of γ. In detail, *CS* is more than 1,000× faster than *CS-Opt* when $\gamma = 0.9$. Moreover, the time costs of *CS* and *CS-Opt* are similar when γ is small. The possible reasons are: (1) Many invalid vertex groups added into Q by *CS-Opt* have small $\mathbb{T}_c$ values, and few of them reach the top of Q before 30 combos of a query vertex are obtained; (2) For some query vertices, the time costs of *CS* and *CS-Opt* are both more than 30 min and recorded as 30 min. Note that the time cost of *CS-Opt* decreases when γ increases from 0.6 to 0.7. The reason may be that with a large γ, few valid neighbors are obtained with *GetValidNeighbors*. Then, fewer vertex groups are generated and processed, and the time cost is reduced.

6.5 Scalability Evaluation

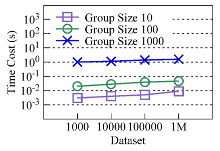
In this subsection, combo searching and $\mathbb{T}$-cohesiveness evolution tracking are performed on the four synthetic datasets to test their scalability. In detail, CF. 1 is applied on these datasets, and $\gamma = 0.5$.

First, the scalability of combo searching is evaluated. Because finding combos on a large graph is time-consuming, for each query vertex, the time cost of processing 100 vertex groups in Q is recorded. For each of the first three datasets, there are 15 query vertices, and for Syn_{1M}, five query vertices are used because of the large time consumption of finding combos in Syn_{1M}. The average time cost on each dataset is reported in Figure 9a. It is shown that the time cost grows almost linearly with the increase of the size of the temporal graph, which suggests that *CS* has reasonable scalability.

Then, the scalability of $\mathbb{T}$-cohesiveness evolution tracking is tested. Specifically, vertex groups of different sizes (10, 100, and 1000) are generated. For each group size, 100 groups of this size are randomly generated, and the total time cost of analyzing these 100 vertex groups is reported. The experimental results are reported in



(a) Combo searching.



(b) $\mathbb{T}$-cohesiveness evolution tracking.

Figure 9: Results of scalability experiments. X-axis labels represent *Syn_{1000}*, *Syn_{10000}*, *Syn_{100000}*, and *Syn_{1M}*, respectively.

Figure 9b. It is shown that the time cost of $\mathbb{T}$-cohesiveness evolution tracking increases slightly with the increase of the size of the temporal graph. It confirms the good scalability of our method.

6.6 Parameter Sensitivity

The parameter sensitivity evaluation is conducted on Email-7d. As shown in Figure 10, CF. 8 is the default configuration, and the T, R and D values are reported when T_1, $T_{0.5}$, R_1, $R_{0.5}$, k, D_1, and $D_{0.5}$ are respectively changed. For each new configuration, there are 30 query vertices, and 30 combos are required for each query vertex.

With the increase of T_1, the Ts of the obtained combos increase. The reason is that when T_1 becomes larger, the TC score of a subgraph with a fixed time span usually increases, and then the $\mathbb{T}$-cohesiveness value of the subgraph increases. It indicates that more subgraphs may be returned as combos, whose Ts are larger.

When the value of $T_{0.5}$ increases, the values of T, R, and D of the obtained combos remain unchanged. The reason may be that the obtained combos have Ts smaller than T_1, and these combos are always obtained when the value of $T_{0.5}$ changes.

When the value of R_1 or $R_{0.5}$ increases, given an original combo $\mathcal{G}_s$, vertices should be added into or removed from $\mathcal{G}_s$, so that $\mathcal{G}_s$ has a larger R value, keeps a high InterTC score, and is still a combo. Since removing vertices from $\mathcal{G}_s$ makes it either less cohesive or become another original combo, new vertices should be added into $\mathcal{G}_s$, and the values of T increase as well.

With the increase of k, according to Equation 5, the density (i.e. D) of a subgraph usually decreases, and the IntraTC score also decreases. In order to make the Ds decrease slightly and get a large IntraTC score, the obtained combos contain more vertices to make their vertices have larger degrees. Then, the Ts and Rs of the combos also increase since more vertices and temporal edges are included.

When the values of D_1 or $D_{0.5}$ increase, the constraint of intra-topological cohesiveness becomes stricter, and the IntraTC score of a subgraph with a fixed density usually decreases. Therefore, combos contain more vertices to be denser and more cohesive in topology. As a result, the time spans of the combos increase.

More sensitivity experiments are also conducted on the other real-world datasets with CF. 1–CF. 28 in Table 3. The results and conclusions are consistent with those presented above, and then are omitted because of the length limitation.

6.7 Case Study

In this subsection, the two proposed time-topology analysis methods are applied on ***DBLP*** to demonstrate their good performances. First, combo searching is performed. In detail, all the temporal edges with timestamps between year 2006 and 2009 are extracted from ***DBLP*** to generate a new temporal graph $\mathcal{G}_0$. Then, the combo searching is performed on $\mathcal{G}_0$ with CF. 22, and γ is set to 0.5. The query vertex is *Jiawei Han*. Figure 11a is one of the obtained combos, and its value of $\mathbb{T}$-cohesiveness is 0.50. The timestamps of temporal edges in Figure 11a are all 2009 (and omitted), which indicates these researchers only coauthored in 2009 during the period [2006, 2009].

The 12 researchers in Figure 11a seem to have cohesive relationships in [2006, 2009] in reality. In detail, *Yizhou Sun*, *Duo Zhang*, *Tianyi Wu*, *Chen Chen*, *Yintao Yu*, *Bo Zhao*, and *Cindy Xide Lin* were students in the Data and Information Systems Research Laboratory in UIUC (abbr. DAIS) in 2008–2009, while *Jiawei Han* and *Chengxiang Zhai* are professors in DAIS. Moreover, *Matt Fredrikson* and his mentor in IBM, i.e. *Mihai Christodorescu*, worked for IBM in 2008 and 2007–2013, respectively. Therefore, since DAIS established a collaboration with IBM from Sept., 2008 to Feb., 2010, these researchers may participate in this research project and coauthor papers in 2009. Then, they had cohesive relationships. Besides, *Binbin Liao* was also a student in UIUC in 2008–2014. He is not a member of DAIS or IBM, and the reason for his cohesive relationships with the other 11 researchers may be that he coauthored with some of these researchers a demo paper for the project of DAIS and IBM. Therefore, it seems these 12 researchers form a combo due to a project, and they are densely connected in a short period of time because this project only lasted for about one and a half years.

To perform quantitative analysis, we also find cohesive subgraphs of *"Jiawei Han"* with four baselines, i.e., Dynamo, KCA$_3$, KCA$_4$, and EquiTruss [1]. Specifically, for EquiTruss, 5-truss communities containing *"Jiawei Han"* are queried similar to [18]. Seven quantitative metrics are used, i.e., d_r, d_t, O_v, deg_{avg}, $|V_s|$, D, and $\mathbb{T}_c$. In detail, for each obtained combo $\mathcal{G}_s = (V_s, \mathcal{E}_s)$, d_r is the maximum distance between any two vertices in $\mathcal{G}_s$; d_t is the maximum difference between two timestamps in $\mathcal{G}_s$; O_v is the number of temporal edges outside $\mathcal{G}_s$, with timestamps in the time period of $\mathcal{G}_s$, and each of the temporal edges is adjacent to a vertex in V_s; deg_{avg} is the average number of neighbors a vertex in V_s has in $\mathcal{G}_s$. The results are shown in Table 5. Note that EquiTruss finds two combos including *"Jiawei Han"* with 13 and 10,955 vertices respectively. We report the statistics of the combo with 13 vertices because it is more reasonable. The experimental results confirm that the combo obtained by our method is the best on almost all the metrics, and is more cohesive than those obtained by the baselines. The combo obtained by KCA$_4$ has a larger deg_{avg}, because it only focuses on the degrees of vertices. As a result, the combo obtained by KCA$_4$ has poor performances on the other metrics such as d_r, O_v, and $\mathbb{T}_c$.

Next, $\mathbb{T}$-cohesiveness evolution tracking is performed on ***DBLP***. The queried vertex group consists of the 12 researchers in Figure 11a, and every time window with a time span of 3 years is traversed. The results are shown in Figure 11b. Because there are no edges among these researchers before 2002, the first time window used is [1999, 2002]. Figure 11b suggests these 12 researchers are cohesive only in window [2006, 2009]. For the time windows before [2006, 2009], as some researchers were not in DAIS, and the project had not started yet, these researchers are not cohesive in the topology dimension. The $\mathbb{T}$-cohesiveness values of these 12 researchers are not 0 in the first four time windows, because *Jiawei Han* and *Chengxiang Zhai* coauthored a paper in 2002. Besides, for the time windows after [2006, 2009], the $\mathbb{T}$-cohesiveness values are small due to three

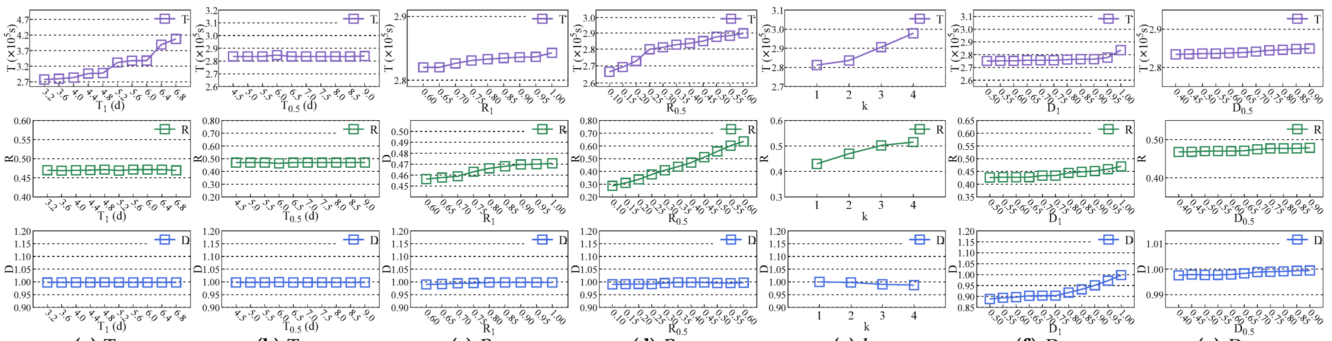
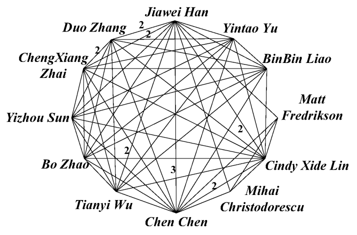
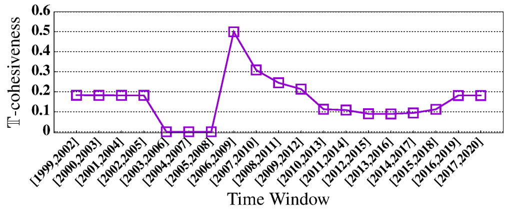

(a) T_1 **(b)** $T_{0.5}$ **(c)** R_1 **(d)** $R_{0.5}$ **(e)** k **(f)** D_1 **(g)** $D_{0.5}$

Figure 10: The results of parameter experiments on Email-7d.



(a) A combo of *Jiawei Han* in [2006,2009]. The number on an edge e is the number of temporal edges corresponding to e, which is 1 by default. Besides, these temporal edges all have timestamps of 2009.



(b) The result of $\mathbb{T}$-cohesiveness evolution tracking on the vertex group in Figure 11a. The $\mathbb{T}$-cohesiveness values are 0 in [2003, 2006], [2004, 2007], and [2005, 2008], because there is no temporal edge in the induced subgraphs of the vertex group in these time periods.

Figure 11: Case Study on DBLP.

possible reasons: (1) *Binbin Liao* and the researchers in IBM do not coauthor with the researchers in DAIS after the project finished; (2) Some researchers of DAIS graduated, and no longer coauthor with the others; (3) The researchers left in DAIS coauthor papers in many years, and they are not cohesive in the time dimension.

7 RELATED WORK

The idea of time-topology analysis is partially inspired by the Time-Frequency Analysis [5] in signal processing. To the best of our knowledge, this paper is the first work to analyze a temporal graph in both the time and topology dimensions.

For $\mathbb{T}$-cohesiveness evolution tracking, there are some previous works that track the evolution of cohesive subgraphs [22, 38, 46]. However, the temporal information is not utilized in these existing methods when the subgraphs are evaluated in each snapshot.

For combo searching, community detection and community search algorithms are proposed to find topologically cohesive subgraphs. Community detection algorithms [2, 14, 15, 23, 32, 40] focus on the global information and divide a graph into cohesive subgraphs. In contrast, community search algorithms [3, 6, 12, 13, 19, 26, 28, 36, 39, 42, 43] focus on the local information and find cohesive subgraphs containing specific vertices. Specifically, a subgraph is considered cohesive if it satisfies a specific structural constraint such as k-clique [10], k-core [7, 13, 21, 35], and k-truss [4, 18].

The above algorithms all find cohesive subgraphs in a static graph, and focus on the topological cohesiveness. There are also some methods proposed for finding cohesive subgraphs in temporal graphs. A typically related problem is dynamic community detection that finds communities in temporal graphs. However, the existing dynamic community detection algorithms [11, 16, 46] only search for densely connected subgraphs on each network snapshot [33], which may lose the topological information in the other snapshots. Besides, the temporal information is only exploited to divide a temporal graph into snapshots, and is not properly utilized.

Table 5: Quantitative comparison of *CS* and the baselines

Alg.	Quantitative Metrics						
	d_r	d_t	O_v	deg_{avg}	$\lvert V_s \rvert$	D	$\mathbb{T}_c$
CS	**2**	**0y**	**37**	8.67	**12**	**1.0**	**0.50**
Dynamo	19	3y	3,051	4.91	2,990	0.93	0.27
KCA$_3$	3	3y	90,150	7.99	164,971	1.0	0.44
KCA$_4$	24	3y	151,703	**9.31**	116,126	1.0	0.36
EquiTruss	2	2y	121	7.69	13	1.0	0.24

8 CONCLUSIONS

In this paper, $\mathbb{T}$-cohesiveness is first proposed to evaluate the cohesiveness of temporal graphs in both the topology and time dimensions. Then, two analysis methods, i.e., $\mathbb{T}$-cohesiveness evolution tracking and combo searching, are proposed. Moreover, an optimization method is proposed to prune the vertex groups whose supergraphs can never be a combo. Finally, experimental results demonstrate that our proposed time-topology analysis methods are superior to Dynamo and k-core algorithms. In the future, we will consider utilizing $\mathbb{T}$-cohesiveness to perform more kinds of time-topology analysis on temporal graphs.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (No. 61872207) and Baidu Inc. Chaokun Wang is the corresponding author.

REFERENCES

[1] Esra Akbas and Peixiang Zhao. 2017. Truss-based community search: a truss-equivalence based indexing approach. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1298–1309.

[2] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (Oct. 2008), P10008.

[3] Lu Chen, Chengfei Liu, Kewen Liao, Jianxin Li, and Rui Zhou. 2019. Contextual community search over large social networks. In *ICDE*. IEEE, 88–99.

[4] Jonathan Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. *National security agency technical report* 16 (2008), 3–29.

[5] Leon Cohen. 1995. *Time-frequency analysis*. Vol. 778. Prentice hall. 70–81 pages.

[6] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yiqi Lu, and Wei Wang. 2013. Online search of overlapping communities. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data*. 277–288.

[7] Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. 2014. Local search of communities in large graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 991–1002.

[8] TA Dang and Emmanuel Viennet. 2012. Community detection based on structural and attribute similarities. In *International conference on digital society (icds)*. 7–12.

[9] Maximilien Danisch, Oana Balalau, and Mauro Sozio. 2018. Listing k-cliques in sparse real-world graphs. In *Proceedings of the 2018 World Wide Web Conference*. 589–598.

[10] Maximilien Danisch, Oana Balalau, and Mauro Sozio. 2018. Listing k-cliques in sparse real-world graphs. In *Proceedings of the 2018 World Wide Web Conference*. 589–598.

[11] Zeineb Dhouioui and Jalel Akaichi. 2014. Tracking dynamic community evolution in social networks. In *ASONAM 2014*. IEEE, 764–770.

[12] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2020. A survey of community search over big graphs. *The VLDB Journal* 29, 1 (2020), 353–392.

[13] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and efficient community search over large heterogeneous information networks. *Proceedings of the VLDB Endowment* 13, 6 (2020), 854–867.

[14] S. Fortunato and D. Hric. 2016. Community detection in networks: A user guide. *Physics Reports* 659 (Nov. 2016), 1–44.

[15] M. Girvan and M. E. J. Newman. 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* 99, 12 (June 2002), 7821–7826. https://doi.org/10.1073/pnas.122653799

[16] Chonghui Guo, Jiajia Wang, and Zhen Zhang. 2014. Evolutionary community structure discovery in dynamic weighted networks. *Physica A: Statistical Mechanics and its Applications* 413 (2014), 565–576.

[17] Wentao Han, Youshan Miao, Kaiwei Li, Ming Wu, Fan Yang, Lidong Zhou, Vijayan Prabhakaran, Wenguang Chen, and Enhong Chen. 2014. Chronos: a graph engine for temporal graph analysis. In *Proceedings of the Ninth European Conference on Computer Systems*. 1–14.

[18] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying k-truss community in large and dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1311–1322.

[19] Xin Huang, Laks VS Lakshmanan, and Jianliang Xu. 2019. *Community search over big graphs*. Vol. 14. Morgan & Claypool Publishers. 1–206 pages.

[20] Caiyan Jia, Yafang Li, Matthew B Carson, Xiaoyang Wang, and Jian Yu. 2017. Node attribute-enhanced community detection in complex networks. *Scientific Reports* 7, 1 (2017), 1–15.

[21] Wissam Khaouid, Marina Barsky, Venkatesh Srinivasan, and Alex Thomo. 2015. K-core decomposition of large networks on a single PC. *Proceedings of the VLDB Endowment* 9, 1 (2015), 13–23.

[22] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2–es.

[23] I. X. Y. Leung, H. Pan, P. Lio, and J. Crowcroft. 2009. Towards real-time community detection in large networks. *Physical review. E, Statistical, nonlinear, and soft matter physics* 79, 6 Pt 2 (June 2009), 066107.

[24] Michael Levi and Peter Reuter. 2006. Money laundering. *Crime and Justice* 34, 1 (2006), 289–375.

[25] Rong-Hua Li, Jiao Su, Lu Qin, Jeffrey Xu Yu, and Qiangqiang Dai. 2018. Persistent community search in temporal networks. In *ICDE*. IEEE, 797–808.

[26] Qing Liu, Yifan Zhu, Minjun Zhao, Xin Huang, Jianliang Xu, and Yunjun Gao. 2020. VAC: Vertex-Centric Attributed Community Search. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 937–948.

[27] Yunkai Lou, Chaokun Wang, Tiankai Gu, Hao Feng, Jun Chen, and Jeffrey Xu Yu. 2021. *Time-Topology Analysis (Full Version)*. Technical Report. Tsinghua University, Beijing, China.

[28] Jiehuan Luo, Xin Cao, Xike Xie, Qiang Qu, Zhiqiang Xu, and Christian S Jensen. 2020. Efficient Attribute-Constrained Co-Located Community Search. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1201–1212.

[29] Pietro Panzarasa, Tore Opsahl, and Kathleen M Carley. 2009. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *Journal of the American Society for Information Science and Technology* 60, 5 (2009), 911–932.

[30] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. 2017. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 601–610.

[31] David H Pyle. 1999. Bank risk management: theory. In *Risk Management and regulation in banking*. Springer, 7–14.

[32] U. N. Raghavan, R. Albert, and S. Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E* 76, 3 (Sept. 2007), 036106.

[33] Giulio Rossetti and Rémy Cazabet. 2018. Community discovery in dynamic networks: a survey. *ACM Computing Surveys (CSUR)* 51, 2 (2018), 1–37.

[34] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. 4292–4293. http://networkrepository.com

[35] Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks* 5, 3 (1983), 269–287.

[36] Mauro Sozio and Aristides Gionis. 2010. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 939–948.

[37] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S Yu. 2007. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 687–696.

[38] Mansoureh Takaffoli, Farzad Sangi, Justin Fagnan, and Osmar R Zäiane. 2011. Community evolution mining in dynamic social networks. *Procedia-Social and Behavioral Sciences* 22 (2011), 49–58.

[39] Chaokun Wang and Junchao Zhu. 2019. Forbidden nodes aware community search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 758–765.

[40] Meng Wang, Chaokun Wang, Jeffrey Xu Yu, and Jun Zhang. 2015. Community detection in social networks: an in-depth benchmarking study with a procedure-oriented framework. *Proceedings of the VLDB Endowment* 8, 10 (2015), 998–1009.

[41] Xiao Wang, Di Jin, Xiaochun Cao, Liang Yang, and Weixiong Zhang. 2016. Semantic community identification in large attribute networks. In *Thirtieth AAAI Conference on Artificial Intelligence*. 265–271.

[42] Zhuo Wang, Weiping Wang, Chaokun Wang, Xiaoyan Gu, Bo Li, and Dan Meng. 2019. Community focusing: yet another query-dependent community detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 329–337.

[43] Yubao Wu, Ruoming Jin, Jing Li, and Xiang Zhang. 2015. Robust local community detection: on free rider effect and its elimination. *Proceedings of the VLDB Endowment* 8, 7 (2015), 798–809.

[44] Zhonggang Wu, Zhao Lu, and Shan-Yuan Ho. 2016. Community detection with topological structure and attributes in information networks. *TIST* 8, 2 (2016), 1–17.

[45] Chen Zhe, Aixin Sun, and Xiaokui Xiao. 2019. Community detection on large complex attribute network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2041–2049.

[46] Di Zhuang, Morris J Chang, and Mingchen Li. 2019. DynaMo: Dynamic community detection by incrementally maximizing modularity. *TKDE* (2019), 1934–1945.