The VLDB Journal (2023) 32:815–843
https://doi.org/10.1007/s00778-022-00772-y

REGULAR PAPER

Time-topology analysis on temporal graphs

Yunkai Lou[1] · Chaokun Wang[1] · Tiankai Gu[1] · Hao Feng[1] · Jun Chen[2] · Jeffrey Xu Yu[3]

Received: 23 February 2022 / Revised: 7 July 2022 / Accepted: 2 December 2022 / Published online: 6 January 2023

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 features, i.e., the temporal feature and topological feature, where the temporal feature is related to the establishing time of the relationships in the temporal graph, and the topological feature is influenced by the structure of the graph. In this paper, considering both these two types of features, we perform time-topology analysis on temporal graphs to analyze the cohesiveness of temporal graphs and extract cohesive subgraphs. Firstly, a new metric named $\mathbb{T}$-cohesiveness is proposed to evaluate 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 cohesive subgraphs containing the query vertex, which have $\mathbb{T}$-cohesiveness values larger than a given threshold. Moreover, since combo searching is NP-hard, a pruning strategy is proposed to estimate the upper bound of the $\mathbb{T}$-cohesiveness value, and then improve the efficiency of combo searching. Experimental results demonstrate the efficiency of the proposed time-topology analysis methods and the pruning strategy. Besides, four more definitions of $\mathbb{T}$-cohesiveness are compared with our method. The experimental results confirm the superiority of our definition.

Keywords Temporal graph · Time-topology analysis · Cohesiveness · Optimization

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 [19,28]. 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 is the relationships among the vertices, and the latter reflects the timestamps when the relationships are established [28]. Then, it is of great importance to analyze the temporal graphs, so that noteworthy information and substructures can be extracted.

There have been many methods for network/graph analysis, and they can be mainly divided into three classes: For static graphs, various community detection [3,16,26,38,46] and community search [21,24,31,33,42,48] algorithms are proposed to find dense subgraphs; For attributed graphs, algo-

✉ Chaokun Wang
chaokun@tsinghua.edu.cn

Yunkai Lou
louyk18@mails.tsinghua.edu.cn

Tiankai Gu
gtk18@tsinghua.org.cn

Hao Feng
fh20@mails.tsinghua.edu.cn

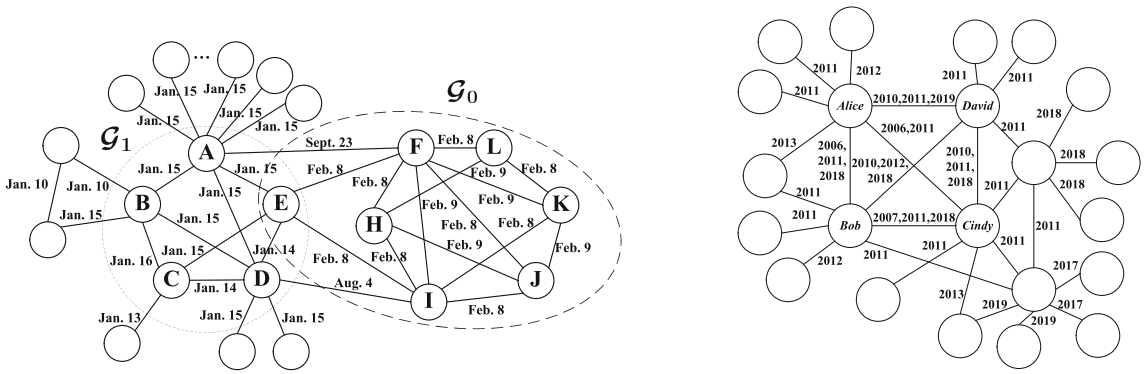
Jun Chen
chenjun22@baidu.com

Jeffrey Xu Yu
yu@se.cuhk.edu.hk

1 School of Software, Tsinghua University, Beijing 100084, China

2 Baidu Inc., Beijing, China

3 The Chinese University of Hong Kong, Hong Kong, China



(a) A money transfer network. (b) A coauthorship network.

Fig. 1 Two sample temporal graphs

rithms are proposed to find cohesive subgraphs according to the network topology and vertex attributes [9,22,47,49,51]; For temporal graphs, existing methods [12,43,52] 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 Fig. 1a, 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) the transactions among these accounts occur in a short period of time (short-time); (2) these accounts have few relationships with the accounts out of the group (cohesion); (3) many transactions occur among these accounts (density). The obtained cohesive accounts can help in many practical applications such as money laundering detection [23,27,29,44] and risk management [37].

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 [24], 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 timestamps of the transactions among them differ 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). 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 the 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 [24] and satisfies Requirement 3 (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 to 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 cohesion and the density together. Similar are the other existing methods.

Another intuitive idea is to firstly find the candidate communities containing vertex E based on the topology (i.e., without considering the timestamps), and then to return the qualified communities by removing the temporal edges with the maximum and minimum timestamps iteratively. However, it is too difficult to achieve the qualified communities since the metrics may change variously when one temporal edge is removed.

Example 2 Given a coauthorship network shown in Fig. 1b, the vertices represent researchers, and an edge between two researchers means that these two researchers have coauthored a paper. Moreover, each edge has a timestamp representing the publication year of the paper. Given four researchers, i.e., Alice, Bob, Cindy, and David, it is interesting to track how the cohesiveness of their relationships changes in every n years (saying $n = 5$).

One basic idea is to evaluate the relationships among them with the k-core structure [24,41] in every n years, and these researchers are considered cohesive if they form a k-core. Suppose $k = 3$. Because these four researchers form a 3-core in time windows [2006, 2010] and [2009, 2013], they are considered to have cohesive relationships in these two time periods. However, in [2006, 2010], David only coauthored papers with the other three researchers in 2010,

while the other researchers coauthored with each other before 2007. It suggests that these four researchers do not coauthor papers together in a short period of time. In [2009, 2013], these four researchers coauthored much more papers with other researchers except themselves. Therefore, these four researchers are actually not cohesive in these two time periods.

The ideal result in Fig. 1a is the induced subgraph of vertices E, F, H, I, J, K, and L, i.e., $\mathcal{G}_0$. Also, a time period when Alice, Bob, Cindy, and David are cohesive in Fig. 1b is [2016, 2020]. To find out them effectively, a new technique named time-topology analysis is devised in this study to discover a group of vertices that have close relationships in both time and topology dimensions (i.e., form a cohesive subgraph).

In our proposed time-topology analysis, the above three requirements in Example 1 are all considered, and the three requirements correspond to the cohesiveness in three perspectives. For a qualified temporal subgraph, its edges should have close timestamps (temporal cohesiveness). Besides, its vertices should have few connections with vertices out of the temporal subgraph (inter-topological cohesiveness), and its vertices should be densely connected (intra-topological cohesiveness). Clearly, as shown in Fig. 1a, the ideal subgraph cannot be found just according to one of the requirements. For example, based on Requirement 3 (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 1 and 2 should also be considered to achieve the satisfactory result.

Next, given a temporal subgraph $\mathcal{G}_s$, three scores (i.e., ζ_t, ζ_{inter}, and ζ_{intra}) are defined to measure the temporal cohesiveness, inter-topological cohesiveness, and intra-topological cohesiveness of $\mathcal{G}_s$, separately. Then, the overall cohesiveness of $\mathcal{G}_s$ (denoted as $\mathbb{T}$-cohesiveness, and abbr. $\mathbb{T}_c$) is proposed, which is the multiplication of the three scores. The reason for multiplying ζ_t, ζ_{inter}, and ζ_{intra} rather than summing them up is that a cohesive subgraph should be cohesive from the perspectives of all the three kinds of cohesiveness, i.e., have large ζ_t, ζ_{inter}, and ζ_{intra} values at the same time. When the cohesiveness of $\mathcal{G}_s$ is computed with summation, a subgraph with a quite small value of ζ_t, ζ_{inter}, or ζ_{intra} may have a large value of cohesiveness and be considered cohesive. By using multiplication, only when the ζ_t, ζ_{inter}, and ζ_{intra} values are all large, a subgraph can have a large value of cohesiveness and is considered cohesive. Therefore, multiplication is utilized.

As shown in Fig. 2, the results of an ablation study demonstrate that each kind of cohesiveness is important. Specifically, $\mathbb{T}_c$ (the first bars of the six subfigures in Fig. 2) represents the method we propose, and it considers all the three kinds of cohesiveness. $\mathbb{T}_c$-ζ_t (the second bars), $\mathbb{T}_c$-ζ_{inter} (the third bars), and $\mathbb{T}_c$-ζ_{intra} (the fourth bars) are the methods

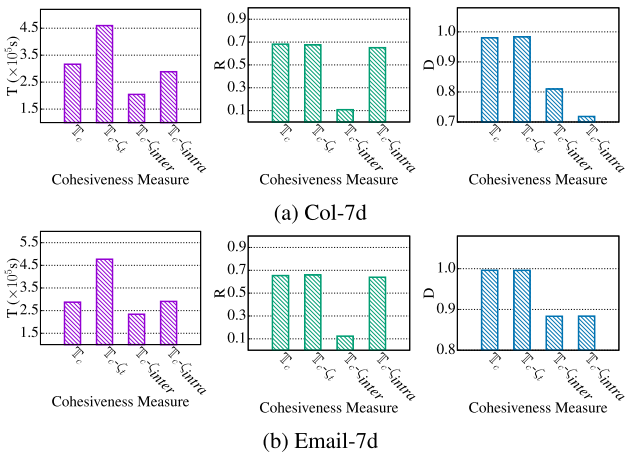


Fig. 2 Results of one ablation study on different cohesiveness measurements. Given a temporal graph $\mathcal{G}$, for a cohesive subgraph $\mathcal{G}_s = (V_s, \mathcal{E}_s)$ of $\mathcal{G}$, T is the time span of $\mathcal{G}_s$, R is the ratio of $|\mathcal{E}_s|$ to the number of temporal edges (in $\mathcal{G}$) adjacent to V_s, and D is the density of $\mathcal{G}_s$. Subgraphs with smaller T, larger R, and larger D values are more cohesive

ignoring ζ_t (temporal cohesiveness), ζ_{inter} (inter-topological cohesiveness), and ζ_{intra} (intra-topological cohesiveness) , respectively. When $\mathbb{T}_c$-ζ_t is used to find cohesive subgraphs, the obtained subgraphs have much larger Ts (shown in the first figures in Fig. 2a, b), which indicates that the obtained subgraphs have much longer time spans and are not cohesive in the time dimension. When $\mathbb{T}_c$-ζ_{inter} is used, the obtained subgraphs have quite small values of R ($R \approx 0.1$) as shown in second figures in Fig. 2a, b, and they have much more connections with vertices out of the subgraphs. When $\mathbb{T}_c$-ζ_{intra} is used , the obtained subgraphs have small values of Ds as shown in the third figures in Fig. 2a, b and are sparse as well as not closely connected. More details can be found in Sect. 6.5. Then, $\mathbb{T}_c$ is used in time-topology analysis to evaluate the cohesiveness of temporal subgraphs.

The main contributions of this paper are 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 features are considered.
(2) We propose a new evaluation metric named Temporal and Topological Cohesiveness (abbr. $\mathbb{T}$-cohesiveness) to evaluate the cohesiveness of temporal subgraphs. $\mathbb{T}$-cohesiveness takes both temporal cohesiveness and topological cohesiveness of a temporal subgraph into account. Based on $\mathbb{T}$-cohesiveness, two typical time-topology analysis methods are proposed, i.e., $\mathbb{T}$-cohesiveness evolution tracking and combo searching.
(3) We prove that combo searching is NP-hard and proposes an efficient optimization method to accelerate

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 ζ_t, ζ_{inter} and ζ_{intra}

the process of combo searching. Specifically, the upper bound of the $\mathbb{T}$-cohesiveness values of a given vertex group's supergroups is computed, so that numerous vertex groups can be pruned in advance. The correctness of the optimization method is also proved.

(4) Four more definitions of $\mathbb{T}$-cohesiveness are proposed and compared with our method in the experiments. The experimental results confirm the superiority of our definition. Besides, the experimental results demonstrate the efficiency and good scalability of our proposed methods. The ablation study also shows the good performance of the proposed optimization method.

The remainder of the paper is organized as follows. Section 2 gives the definitions of some basic concepts used in this paper. Section 3 proposes $\mathbb{T}$-cohesiveness and presents the algorithm to compute the $\mathbb{T}$-cohesiveness of a temporal subgraph. Section 4 presents two useful time-topology analysis methods on temporal graphs, i.e., $\mathbb{T}$-cohesiveness evolution tracking and combo searching. Section 5 proposes to prune the temporal subgraphs whose supergroups cannot be cohesive enough to realize more efficient analysis. Section 6 reports the experiments. Section 7 reviews the related work. Section 8 concludes this paper.

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. For convenience, the minimal timestamp in $\mathcal{G}$ is denoted as $t_{\min}$, the maximal timestamp is denoted as $t_{\max}$, and the time span of $\mathcal{G}$ is $t_{\max} - t_{\min}$. A subgraph of $\mathcal{G}$ is called a temporal subgraph.

Example 3 A temporal graph is shown in Fig. 3 and has timestamps on its edges. 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 $I_{\mathcal{G}}(V_s) = (V_s, \mathcal{E}_s)$ (abbr. $I(V_s)$ when there is no ambiguity), where $\mathcal{E}_s = \{(u, v, t) | u, v \in V_s, (u, v, t) \in \mathcal{E}\}$.

Example 4 The induced subgraph of $V_s = \{v_2, v_3, v_4\}$ on the graph in Fig. 3 is $I(V_s) = (V_s, \mathcal{E}_s)$, where $\mathcal{E}_s = \{(v_2, v_3, 98), (v_2, v_3, 100), (v_2, v_4, 97), (v_3, v_4, 98), (v_3, v_4, 99)\}$.

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 $\mathcal{G} = (V, \mathcal{E})$, its projected graph is denoted as $G^P = (V^P, E^P)$, where $E^P = \{(u, v) | (u, v, t) \in \mathcal{E}\}$. Each edge $(u, v) \in E^P$ is called a normal edge.

Example 5 The projected graph G^P of the temporal graph in Fig. 3 is shown in Fig. 4a.

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 if and only if

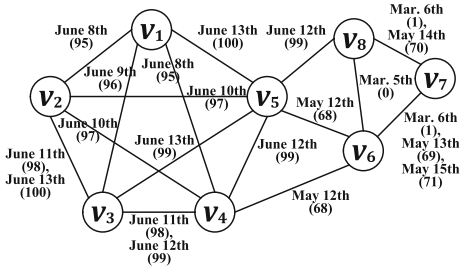
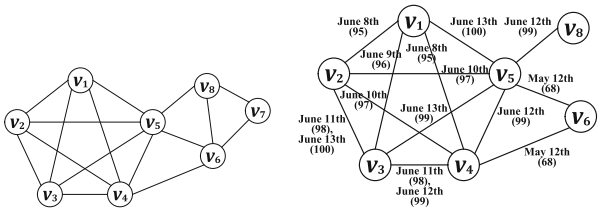


Fig. 3 A sample temporal graph. The date on each edge represents the day the interaction happens, and the number in the parenthesis is the corresponding timestamp



(a) The projected graph of Fig. 3. (b) The local structure of $\{v_1, v_2, v_3, v_4, v_5\}$ in Fig. 3.

Fig. 4 Temporal graphs used to explain the definitions

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.

Example 6 Given the temporal graph in Fig. 3 (denoted as $\mathcal{G}$), let $V_s = \{v_1, v_2, v_3, v_4, v_5\}$. Then, $\mathcal{G}_s = I_{\mathcal{G}}(V_s)$ is a combo. Firstly, the timestamps of the temporal edges in $\mathcal{G}_s$ are in a short period of time (5 days), which indicates that $\mathcal{G}_s$ is temporally cohesive. Secondly, vertices in $\mathcal{G}_s$ are densely connected (a complete graph with five vertices), and most temporal edges adjacent to vertices in V_s are inside $\mathcal{G}_s$, so $\mathcal{G}_s$ is topologically cohesive. Therefore, $\mathcal{G}_s$ is a combo. However, $\mathcal{G}_1 = I_{\mathcal{G}}(\{V_s \cup \{v_6\}\})$ is not a combo, because its time span is longer than a month and $\mathcal{G}_1$ is not cohesive in the time dimension. Moreover, $\mathcal{G}_2 = I_{\mathcal{G}}(\{V_s \cup \{v_8\}\})$ is not a combo, because v_8 is not densely connected to the other vertices in $\mathcal{G}_2$, and $\mathcal{G}_2$ is not topologically cohesive.

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. Next, the concept of a local structure is proposed, and it is later used to evaluate the topological cohesiveness of a given temporal subgraph.

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.

Example 7 Given Fig. 3 and $V_s = \{v_1, v_2, v_3, v_4, v_5\}$, the local structure of V_s is shown in Fig. 4b.

3 $\mathbb{T}$-cohesiveness

In this section, the main idea of Temporal and Topological Cohesiveness ($\mathbb{T}$-cohesiveness) is first proposed. Then, the three terms of $\mathbb{T}$-cohesiveness, i.e., ζ_t, ζ_{inter}, and ζ_{intra}, are detailed. Finally, 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 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 (abbreviated as $\mathbb{T}$-cohesiveness) to evaluate temporal subgraphs. $\mathbb{T}$-cohesiveness takes both temporal and topological information of a temporal graph into consideration and can evaluate temporal subgraphs efficiently.

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. 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 value of $\mathbb{T}_c$ indicates a more cohesive subgraph. Please note that the $\mathbb{T}$-cohesiveness of a vertex group V_s means the $\mathbb{T}$-cohesiveness of $I_G(V_s)$ in this paper. 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:

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

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 Eq. 2. Besides, $\max(T, T_1)$ is used to guarantee 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 in T, as shown in Fig. 5. (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.

The parameters T_1 and $T_{0.5}$ correspond to the thresholds of perfect and acceptable time spans of subgraphs, respectively.

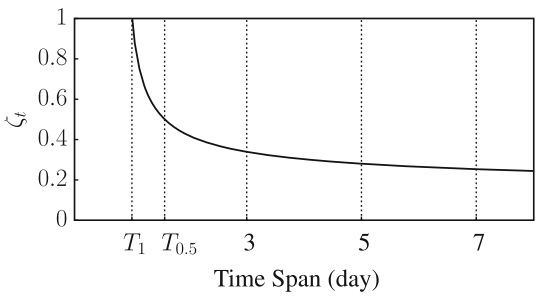


Fig. 5 An example of the curve of Eq. 2. Note that the TC scores of the temporal graphs with the time spans of T_1 and $T_{0.5}$ are 1 and 0.5, respectively

Leaving them 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 perfectly temporally cohesive, and we can set the perfect time span $T_1 = 0$ year according to Definition 1. However, in a temporal graph representing money transfer, a perfectly temporally cohesive subgraph should have a much shorter time span such as a week. Then, we can set the perfect time span $T_1 = 6$ days to evaluate the temporal cohesiveness of subgraphs in this temporal graph. Therefore, T_1 and $T_{0.5}$ can ensure the flexibility of the temporal cohesiveness score.

Note that, the suitable values of T_1 and $T_{0.5}$ are subjective, and they should be specified according to the practical conditions and users' preferences. Our suggestions on how to specify the values of T_1 and $T_{0.5}$ are as follows: Firstly, the value of T_1 should satisfy that subgraphs with time spans shorter than or equal to T_1 are perfectly cohesive in the time dimension in the corresponding condition. Secondly, the value of $T_{0.5}$ should satisfy that subgraphs with time spans shorter than or equal to $T_{0.5}$ are acceptable to be considered cohesive in the time dimension, and it is a loose version of the perfect constraint w.r.t. T_1. The methods to set the values of the parameters of ζ_{inter} and ζ_{intra} are similar and omitted.

Example 8 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 equal to or less than 4 days, the TC score is 1, and if they contact in a time span of 7 days, the TC score is 0.5. As shown in Fig. 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 5 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 $R(\mathcal{G}_s) = \frac{|\mathcal{E}_s|}{|\mathcal{E}_s^l|}$, 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 as follows:

$$\zeta_{\text{inter}}(\mathcal{G}_s) = \begin{cases} \frac{1}{1+\log\left(\frac{e-1}{R_1-R_{0.5}} * (R_1 - \min(R(\mathcal{G}_s), R_1)) + 1\right)} & |\mathcal{E}_s| \neq 0 \\ 0 & |\mathcal{E}_s| = 0 \end{cases} \tag{3}$$

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 InterTC scores of 1 and 0.5, respectively, in Eq. 3.

Algorithm 1: computeTC

Input: $\mathcal{G} = (V, \mathcal{E})$: temporal graph, $\mathcal{G}_s = (V_s, \mathcal{E}_s)$: subgraph of $\mathcal{G}$
Output: $\mathbb{T}_c$: $\mathbb{T}$-cohesiveness value of $\mathcal{G}_s$

1 // compute ζ_t
2 $T \leftarrow$ the time span of $\mathcal{G}_s$
3 $\zeta_t \leftarrow \frac{1}{1+log(\frac{e-1}{T_{0.5}-T_1}*(\max(T,T_1)-T_1)+1)}$
4 // compute ζ_{inter}
5 $|\mathcal{E}_s^l| \leftarrow$ the number of temporal edges in the local structure of $\mathcal{G}_s$
6 $R \leftarrow \frac{|\mathcal{E}_s|}{|\mathcal{E}_s^l|}$
7 **if** $R = 0$ **then**
8 $\quad \zeta_{\text{inter}} \leftarrow 0$
9 **else**
10 $\quad \zeta_{\text{inter}} \leftarrow \frac{1}{1+log(\frac{e-1}{R_1-R_{0.5}}*(R_1-\min(R,R_1))+1)}$
11 // compute ζ_{intra}
12 $\mathcal{G}_s^P = (V_s, E_s) \leftarrow$ the projected graph of $\mathcal{G}_s$
13 $total \leftarrow 0$
14 **foreach** *vertex* v *in* V_s **do**
15 $\quad total$ += $\min(k, deg_{\mathcal{G}_s^P}(v))$
16 $D \leftarrow \frac{total}{|V_s|*k}$
17 **if** $D = 0$ **then**
18 $\quad \zeta_{\text{intra}} \leftarrow 0$
19 **else**
20 $\quad \zeta_{\text{intra}} \leftarrow \frac{1}{1+log(\frac{e-1}{D_1-D_{0.5}}*(D_1-\min(D,D_1))+1)}$
21 // compute the value of $\mathbb{T}$-cohesiveness
22 $\mathbb{T}_c \leftarrow \zeta_t * \zeta_{\text{intra}} * \zeta_{\text{inter}}$
23 return $\mathbb{T}_c$

Example 9 Let $R_1 = 0.9$ and $R_{0.5} = 0.4$. In Fig. 3, let $\mathcal{G}_s = I(\{v_1, v_2, v_3, v_4, v_5\})$. $\mathcal{G}_s$ has 12 intra-edges and 3 inter-edges, and its ratio of intra-edges is $\frac{12}{15}$. Then, $\zeta_{\text{inter}}(\mathcal{G}_s) = 0.77$.

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 [10], the value of *Density* is always small, and such definition of *Density* is inefficient in distinguishing densely connected subgraphs in real scenes.

Therefore, k-core [41] is proposed to evaluate the density of a graph. k-core is a structure utilized in community search algorithms [8,24], and for a subgraph that is a k-core, the degree of each vertex in it should be larger than $k-1$. Please note that the degree of a vertex in a temporal graph $\mathcal{G}$ means its degree in G^P. As k-core can evaluate whether vertices are closely engaged in the subgraph, a density function D w.r.t. a temporal graph $\mathcal{G}_s$ is proposed based on the k-core structure as follows:

$$D(\mathcal{G}_s) = 1 - \frac{\sum_{v \in V_s} \max(k - deg_{G_s^P}(v), 0)}{|V_s| * k} = \frac{\sum_{v \in V_s} \min(k, deg_{G_s^P}(v))}{|V_s| * k} \tag{4}$$

where $deg_{G_s^P}(v)$ is the degree of v in G_s^P, and k is the expected minimal degree in the subgraph (i.e., the subgraph is a k-core). $D(\mathcal{G}_s)$ is abbreviated as D when there is no ambiguity. In this definition, $\sum_{v \in V_s} \max(k - deg_{G_s^P}(v), 0)$ is the minimum degree necessary to convert $\mathcal{G}_s$ into a k-core, and the density of $\mathcal{G}_s$ is defined as the ratio of the actual sum of degrees in $\mathcal{G}_s$ (each degree is truncated at k) to the minimum sum of degrees that a k-core with V_s vertices can have. Note that, 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:

$$\zeta_{\text{intra}}(\mathcal{G}_s) = \begin{cases} \frac{1}{1+\log\left(\frac{e-1}{D_1-D_{0.5}}*(D_1-\min(D(\mathcal{G}_s),D_1))+1\right)} & |\mathcal{E}_s| \neq 0 \\ 0 & |\mathcal{E}_s| = 0 \end{cases} \tag{5}$$

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 Eq. 5.

Example 10 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 two neighbors is 0.5. For $\mathcal{G}_s = I(\{v_1, v_2, v_3, v_4, v_5\})$ in Fig. 3, because $\mathcal{G}_s$ is a 4-core, $\zeta_{\text{intra}}(\mathcal{G}_s) = 1$.

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

Algorithm 1 is proposed to compute the $\mathbb{T}$-cohesiveness of a given temporal subgraph $\mathcal{G}_s$. With a carefully designed data structure, e.g., storing the temporal graph with the data structure as follows:

map<VID1, pair<NUM, List<pair<VID2, Timestamp> > > >

The time complexities of computing ζ_t, ζ_{inter}, and ζ_{intra} are all $O(|V_s|)$, and the time complexity of Algorithm 1 is $O(|V_s|)$. Please note that *NUM* is the number of neighbors of vertex *VID1* in the temporal graph, and the pairs $pair < VID2, Timestamp >$ are sorted in the ascending order of the value of *Timestamp*.

Example 11 Given $\mathcal{G}_s = I(\{v_1, v_2, v_3, v_4, v_5\})$ in Fig. 3, $\zeta_t(\mathcal{G}_s) = 0.69$ according to Line 3 of Algorithm 1. Next, $\zeta_{\text{inter}}(\mathcal{G}_s) = 0.77$ according to Line 10, and $\zeta_{\text{intra}} = 1$ according to Line 20. Finally, Line 22 computes the $\mathbb{T}$-cohesiveness of $\mathcal{G}_s$, and $\mathbb{T}_c(\mathcal{G}_s) = 0.53$.

Table 2 Different definitions of $\mathbb{T}$-cohesiveness. Please note that $T(\mathcal{G}_s)$, $R(\mathcal{G}_s)$, and $D(\mathcal{G}_s)$ are abbreviated as T, R, and D

Notation	Cohesiveness	ζ_t	ζ_{inter}	ζ_{intra}
$\mathbb{T}_c$	$\zeta_t * \zeta_{\text{inter}} * \zeta_{\text{intra}}$	$\frac{1}{1+\log\left(\frac{e-1}{T_{0.5}-T_1}(\max(T,T_1)-T_1)+1\right)}$	$\begin{cases} \frac{1}{1+\log\left(\frac{e-1}{R_1-R_{0.5}}(R_1-\min(R,R_1))+1\right)} & \lvert\mathcal{E}_s\rvert \neq 0 \\ 0 & \lvert\mathcal{E}_s\rvert = 0 \end{cases}$	$\begin{cases} \frac{1}{1+\log\left(\frac{e-1}{D_1-D_{0.5}}(D_1-\min(D,D_1))+1\right)} & \lvert\mathcal{E}_s\rvert \neq 0 \\ 0 & \lvert\mathcal{E}_s\rvert = 0 \end{cases}$
$\mathbb{T}_{\text{avg}}$	$\frac{1}{3}\zeta_t + \frac{1}{3}\zeta_{\text{inter}} + \frac{1}{3}\zeta_{\text{intra}}$	$\frac{1}{1+\log\left(\frac{e-1}{T_{0.5}-T_1}(\max(T,T_1)-T_1)+1\right)}$	$\begin{cases} \frac{1}{1+\log\left(\frac{e-1}{R_1-R_{0.5}}(R_1-\min(R,R_1))+1\right)} & \lvert\mathcal{E}_s\rvert \neq 0 \\ 0 & \lvert\mathcal{E}_s\rvert = 0 \end{cases}$	$\begin{cases} \frac{1}{1+\log\left(\frac{e-1}{D_1-D_{0.5}}(D_1-\min(D,D_1))+1\right)} & \lvert\mathcal{E}_s\rvert \neq 0 \\ 0 & \lvert\mathcal{E}_s\rvert = 0 \end{cases}$
$\mathbb{T}_{\text{m}}$	$\zeta_t * \zeta_{\text{inter}} * \zeta_{\text{intra}}$	$\frac{1}{T+1}$	R	D
$\mathbb{T}_{\text{m1}}$	$\zeta_t * \zeta_{\text{inter}} * \zeta_{\text{intra}}$	$\begin{cases} \frac{1}{1+T-T_1} & T \geq T_1 \\ 1 & T < T_1 \end{cases}$	$\min(\frac{R}{R_1}, 1)$	$\min(\frac{D}{D_1}, 1)$
$\mathbb{T}_{\text{mboth}}$	$\zeta_t * \zeta_{\text{inter}} * \zeta_{\text{intra}}$	$\begin{cases} \frac{T_{0.5}-T_1}{T+T_{0.5}-2T_1} & T \geq T_1 \\ 1 & T < T_1 \end{cases}$	$\begin{cases} \frac{0.5R}{R_{0.5}} & R \leq R_{0.5} \\ \frac{0.5(R-R_{0.5})}{R_1-R_{0.5}} + 0.5 & R_{0.5} \leq R \leq R_1 \\ 1 & R > R_1 \end{cases}$	$\begin{cases} \frac{0.5D}{D_{0.5}} & D \leq D_{0.5} \\ \frac{0.5(D-D_{0.5})}{D_1-D_{0.5}} + 0.5 & D_{0.5} \leq D \leq D_1 \\ 1 & D > D_1 \end{cases}$

3.6 Other possible definitions of $\mathbb{T}$-cohesiveness

As shown in Table 2, besides the definition of $\mathbb{T}$-cohesiveness proposed above (i.e., the first line in Table 2), some other possible definitions are presented. These new definitions are generated by either changing the method to combing ζ_t, ζ_{inter}, and ζ_{intra}, or using new definitions of ζ_t, ζ_{inter}, and ζ_{intra}.

Specifically, $\mathbb{T}_{\text{avg}}$ computes the average of ζ_t, ζ_{inter}, and ζ_{intra} as the $\mathbb{T}$-cohesiveness of a combo, instead of multiplying the scores like $\mathbb{T}_c$. Moreover, by utilizing definitions of ζ_t, ζ_{inter}, and ζ_{intra} different from those of $\mathbb{T}_c$, three more definitions, i.e. $\mathbb{T}_{\text{m}}$, $\mathbb{T}_{\text{m1}}$, and $\mathbb{T}_{\text{mboth}}$, are proposed. In detail, in $\mathbb{T}_{\text{m}}$, ζ_t is defined as an anti-monotonic function w.r.t. T, and ζ_{inter} and ζ_{intra} are defined as two monotonic functions w.r.t. R and D, respectively. In order to add more user-specific parameters, $\mathbb{T}_{\text{m1}}$ uses T_1, R_1, and D_1 in the definitions. Furthermore, $\mathbb{T}_{\text{mboth}}$ introduces $T_{0.5}$, $R_{0.5}$, and $D_{0.5}$ in the definitions as well.

Note that, $\mathbb{T}_c$ is more reasonable than $\mathbb{T}_{\text{avg}}$, because multiplying ζ_t, ζ_{inter}, and ζ_{intra} to evaluate the $\mathbb{T}$-cohesiveness of subgraphs can ensure that a cohesive subgraph is cohesive in all the three perspectives, while summing them up cannot. The performances of the five different definitions in Table 2 are compared with experiments in Sect. 6.3, and the experimental results confirm the superiority of $\mathbb{T}_c$.

3.7 $\mathbb{T}$-cohesiveness for directed graphs

Similar to many previous studies [8,20,48], this paper focuses on the $\mathbb{T}$-cohesiveness of the undirected graphs, but this work 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, a directed temporal edge between two accounts represents 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, two users are considered as friends 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 undirected graphs, the $\mathbb{T}$-cohesiveness can be easily applied to evaluate their combos.

Another method to deal with directed graphs is to change the definition of $\mathbb{T}$-cohesiveness. In detail, based on the idea of evaluating temporal graphs from the perspectives of temporal, inter-topological, and intra-topological cohesiveness, we can change the definition of $\mathbb{T}$-cohesiveness for directed graphs according to these three perspectives.

For the temporal cohesiveness, since the time span is unrelated to the directions of the edges, its definition can stay the same.

For the inter-topological cohesiveness of a directed temporal subgraph $\mathcal{G}_s$, the inter-edges of $\mathcal{G}_s$ are the directed edges whose adjacent vertices are both inside $\mathcal{G}_s$, and the intra-edges of $\mathcal{G}_s$ are the directed edges with only one adjacent vertex inside $\mathcal{G}_s$. A temporal subgraph $\mathcal{G}_s$ with a large InterTC score should have much more inter-edges than intra-edges. Moreover, weights may be assigned to the directed edges when the cohesiveness of a directed temporal subgraph is evaluated. For example, given an email network $\mathcal{G}_s$, each directed edge represents that two adjacent persons have contacted through an email, and the source of the edge is the sender of the email. Suppose vertex $u \in \mathcal{G}_s$, and there is an edge from u to a vertex v_i, which is out of $\mathcal{G}_s$. Meanwhile, there is also an edge from v_o to u, where v_o is out of $\mathcal{G}_s$. Then, it seems that the edge from u to v_i and the edge from v_o to u should have different weights, and the weight of the edge from u to v_i may be larger when we evaluate the cohesiveness of $\mathcal{G}_s$. The reason is that for person u, v_i may be one of her/his friends, while v_o may be a stranger or a salesman. It may be difficult but significant to specify the weights of the edges in a directed temporal subgraph. After the weights of edges are specified, the definition of the inter-topological cohesiveness score of a temporal subgraph can be designed accordingly. Note that, when all the edges are weighted as 1, it is the same as our method used in the paper, which considers each directed edge as an undirected edge.

For the intra-topological cohesiveness of a temporal subgraph $\mathcal{G}_s$, in the undirected version of $\mathbb{T}$-cohesiveness, k-core is used to evaluate the density of a temporal subgraph, and k-core is usually applied to find communities on undirected graphs [7,13]. Therefore, when the intra-topological cohesiveness of a directed subgraph is evaluated, k-core may be replaced with its directed version (i.e., D-core [17]).

The above is one way to generalize $\mathbb{T}$-cohesiveness to deal with directed graphs, and designing a simple but efficient directed version of $\mathbb{T}$-cohesiveness will take many efforts. Therefore, we will study it in our future work.

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 (Sect. 4.1) and combo searching (Sect. 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 by. It has practical significance and wide applications such as tracking the evolution of organizations and detecting money launderers.

Algorithm 2: $\mathbb{T}$-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: Γ: the list of the $\mathbb{T}$-cohesiveness of V_s in all the time windows

```
t_s ← t_min, t_e ← t_min+w //the first time window
while t_s < t_max do
    G_t ← extract the temporal subgraph of G with all the edges whose
          timestamps are in [t_s, t_e]
    G_g ← I_{G_t}(V_s)
    T_c ← computeTC(G, G_g)
    Γ.push(T_c)
    t_s ← t_s + st, t_e ← t_s + w
return Γ
```

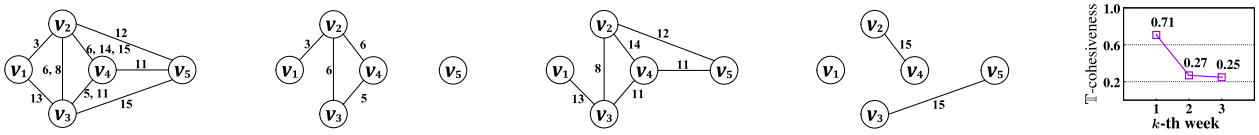
Algorithm 2 shows the process of $\mathbb{T}$-cohesiveness evolution tracking. Specifically, the vertex group is fixed, and the time dimension changes. The time complexity of Algorithm 2 is $O(|\mathcal{E}| + N_w|V_s|)$, where N_w represents the number of windows. Besides, the space complexity of Algorithm 2 is $O(|V_s| + |V| + |\mathcal{E}| + N_w)$.

Example 12 The analysis of $\mathbb{T}$-cohesiveness evolution tracking is illustrated in Fig. 6. Figure 6a is a communication network, where each vertex is a person, and the timestamp on an edge is the day the adjacent persons contact. For example, timestamp "3" on the edge between v_1 and v_2 indicates that these two persons contact on the third day.

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 7 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 Fig. 6e. v_2, v_3, and v_4 are cohesive in the first week (Fig. 6b), because these vertices form a 2-core ($\zeta_{\text{intra}} = 1$), and there is only one inter-edge ($\zeta_{\text{inter}} = 0.71$). Moreover, the timestamps among them are in a short period of time ($\zeta_t = 1$). In the second week (Fig. 6c), 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 ($\zeta_{\text{inter}} = 0.54$). Besides, in the third week (Fig. 6d), v_2, v_3, and v_4 are not cohesive, since the ratio of the intra-edges is still small ($\zeta_{\text{inter}} = 0.54$), and the degrees of these three vertices are at most 1 ($\zeta_{\text{intra}} = 0.46$).

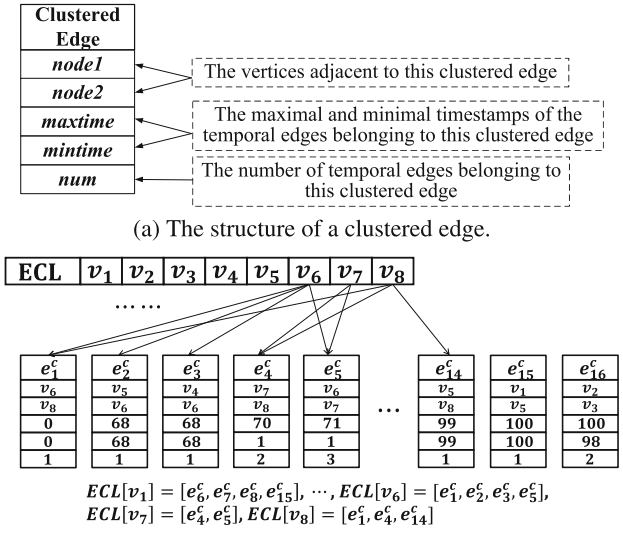
4.2 Combo searching

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 values of $\mathbb{T}$-cohesiveness are at least γ. Different from $\mathbb{T}$-cohesiveness evolution tracking, in the process of 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) changes. In this subsection, we first propose an index called Edge-Clustered



(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).

Fig. 6 An example of performing $\mathbb{T}$-cohesiveness evolution tracking on a temporal graph



(a) The structure of a clustered edge.

(b) The ECL index of Fig. 3.

Fig. 7 The edge-clustered-list index

List to efficiently store the temporal edges (Sect. 4.2.1). Then, the algorithm for combo searching is presented (Sect. 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 Fig. 7.

The basic unit of the ECL index is a clustered edge. A clustered edge 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. Specifically, the structure of a clustered edge e^c is shown in Fig. 7a, and it consists of five items. First, e^c records its adjacent vertices in *node1* and *node2*. Second, *maxtime* and *mintime* represent the maximal timestamp and the minimal timestamp 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 (i.e., 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 the process of 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 necessary 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 in memory to perform combo searching.

Example 13 The ECL index of the temporal graph in Fig. 3 is shown in Fig. 7b, and 16 clustered edges are generated. The clustered edges adjacent to vertex u can be obtained with $ECL[u]$.

4.2.2 Algorithm for combo searching

The *Combo Searching* (abbr. *CS*) algorithm that performs combo searching is detailed in Algorithm 3. Since each combo is an induced subgraph of a vertex group, *CS* first traverses different vertex groups, and then computes the $\mathbb{T}$-cohesiveness values of their induced subgraphs to find combos.

Firstly, the time span that makes the TC score $\zeta_t = \gamma$ is computed at Line 1. Note 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. Then, priority queue Q is initialized. The vertex groups are organized with 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 vertex group with the maximal $\mathbb{T}$-cohesiveness in Q (which is denoted as g)

Algorithm 3: Combo Searching (*CS*)

Input: $\mathcal{G} = (V, \mathcal{E})$: a temporal graph, v: the query vertex, γ: the threshold of $\mathbb{T}$-cohesiveness, n: the number of combos to obtain, *ECL*: the ECL index of $\mathcal{G}$
Output: σ: n combos

1 $maxspan \leftarrow \frac{(e^{\frac{1}{\gamma}-1}-1)*(T_{0.5}-T_1)}{e-1} + T_1$
2 $Q \leftarrow \emptyset$ // Q is a priority queue of vertex groups, where 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 σ
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 σ

is retrieved and processed in each iteration (Lines 5–20). Specifically, *GetValidNeighbors* is invoked to acquire all the valid neighbors of the vertices in g, and these valid neighbors are stored into *validv*, while the invalid neighbors are stored in *invalidv* (Line 7). Suppose u is a neighbor of a vertex in g, then u is called a valid neighbor of vertices in g, if and only if the time span of $I_{\mathcal{G}}(g \cup \{u\})$ is not longer than *maxspan*. Otherwise, u is an invalid neighbor. Note that, the invalid neighbors should never be added into g, because a temporal subgraph containing g and an invalid neighbor of vertices in g cannot be a combo.

For each valid neighbor $u \in validv$, u is added into g to generate a new vertex group *cur* (Line 9). If *cur* has been processed before, it is skipped (Line 10). If *cur* is never processed before, its induced subgraph $\mathcal{G}_q$ is obtained (Line 11), and the $\mathbb{T}$-cohesiveness of $\mathcal{G}_q$ is computed (Line 12). If $\mathcal{G}_q$ has a value of $\mathbb{T}$-cohesiveness larger than the threshold γ, $\mathcal{G}_q$ is a combo and is added into the result list σ (Line 14). Then, *cur* is added into Q, and it can be further extended and processed in the iterations later (Line 20).

The time complexity of *GetValidNeighbors* is $O(|E^P|)$, where $|E^P|$ is the number of edges in $\mathcal{G}$'s projected graph. The best time complexity of Algorithm 3 is $O\left(n(|E^P| + |V|)\right)$, while the worst time complexity is $O\left(2^{|V|}(|E^P| + |V|)\right)$. Moreover, as the space cost of the ECL index is $O(|V| + |E^P|)$, the best space cost of Algorithm 3 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 in a temporal graph is NP-hard.*

Proof First, it is obvious that given a vertex group V_s, checking whether $\mathbb{T}_c(I(V_s)) \geq \gamma$ can be finished in polynomial time. Second, it is noted that the problem of finding a (k_0, r)-core [50] is NP-hard. Given a graph $G = (V, E)$ and a counter C initialized with 1, we construct a new temporal graph $\mathcal{G} = (V, \mathcal{E})$ as follows. For two vertices $u, v \in V$, if u and v are similar and connected in G, $(u, v, 0) \in \mathcal{E}$, where 0 is the timestamp. If u and v are similar and not connected in G, they are still not connected in $\mathcal{G}$. It u and v are dissimilar and connected in G, $(u, v, C.val) \in \mathcal{E}$. If u and v are dissimilar, no matter whether they are connected or unconnected in G, $(u, v, C.val) \in \mathcal{E}$. That is to say, the first edge between two dissimilar vertices has a timestamp of 1, and the n-th such edge has a timestamp of n. Then, the problem of finding a (k_0, r)-core ($k_0 \geq 2$) on G can be reduced to the problem of finding a combo for each vertex on $\mathcal{G}$ with $T_1 = 0$, $R_1 = \epsilon \rightarrow 0$, $k = k_0$, $D_1 = 1$, and $\gamma = 1$. Hence, the problem of finding a combo is NP-hard. □

Since the combo searching problem is NP-hard, we carefully design our *CS* algorithm to speed up, and it 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 to deal with at Line 5, vertex groups with large $\mathbb{T}$-cohesiveness 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 14 Given Fig. 3 as a temporal graph (denoted as $\mathcal{G}$), the *CS* algorithm (Algorithm 3) is conducted on $\mathcal{G}$ 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 $\gamma = 0.5$. Please note that in the following examples, the parameters are also set like this. First, the value of *maxspan* is computed to be 7 days at Line 1. Line 3 initializes the priority queue Q with the query vertex, i.e., $Q = [(\{v_1\}, 0)]$. Then, Lines 4–20 traverse all the possible vertex groups and compute their values of $\mathbb{T}$-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 a new vertex group 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 values of $\mathbb{T}$-cohesiveness are calculated at Line 12. Because their values of $\mathbb{T}$-cohesiveness are all $0.17 < \gamma$, 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 vertex group $\{v_1, v_2, v_3, v_4, v_5\}$ is larger than γ. Therefore, these four new vertex groups are pushed into Q (Line 20) and are 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 $g \subset g'$.

Example 15 Given a temporal graph as shown in Fig. 3 and a vertex group $g = \{v_1, v_2, v_3, v_4\}$, $\{v_1, v_2, v_3, v_4, v_5\}$ and $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ are two supergroups of g.

The relationship of supergroup is transitive. In Algorithm 3, 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 3, every newly obtained vertex group *cur* is inserted into the priority queue Q and processed in the later iterations (Line 20). However, note 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 16 Let $cur = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ be a vertex group in Fig. 3, and $\overline{cur}$ is a supergroup of *cur*. We have *maxspan* = 7 days. The time span of *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 in Algorithm 3.

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 and never processed.

As defined in Eq. 1, $\mathbb{T}$-cohesiveness consists of three terms, i.e., 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, and then, the upper bound of $\mathbb{T}$-cohesiveness values of *cur*'s supergroups can be obtained.

In the following subsections, the methods to compute the values of $\widehat{\zeta}_t$, $\widehat{\zeta}_{\text{inter}}$, and $\widehat{\zeta}_{\text{intra}}$ are first proposed, respectively (Sects. 5.1–5.3). Then, the algorithm to compute the upper bound of the $\mathbb{T}$-cohesiveness values of *cur*'s supergroups is presented (Sect. 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 not 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 17 In Fig. 3, suppose $cur = \{v_1, v_2, v_3, v_4\}$. Because $T(I(cur)) = 5$ days, $\widehat{\zeta}_t = \zeta_t(I(cur)) = 0.69$.

Lemma 1 *Vertex groups with $\widehat{\zeta}_t < \gamma$ are invalid vertex groups.*

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

As shown in Eq. 3, the value of ζ_{inter} is determined by that of R, and the upper bound of ζ_{inter} is determined by the upper bound 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 upper bound of R.

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

For a vertex $v_i \notin cur$, $N_{sound}^{c(v_i)}$ represents the number of the sound clustered edges w.r.t. *cur* adjacent to v_i, and $N_{unsound}^{c(v_i)}$ represents the number of the unsound clustered edges. 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*.

Example 18 Given the temporal graph in Fig. 3 (denoted as $\mathcal{G}$), $cur = \{v_1, v_2, v_3, v_4, v_5\}$ is a vertex group, and *maxspan* = 7 days. For vertex $v_8 \notin cur$, only the clustered edge between v_5 and v_8 is sound w.r.t. *cur*. The clustered edge between v_6 and v_8 violates the vertex constraint because v_6

is an invalid neighbor of vertices in *cur*. The clustered edge between v_7 and v_8 violates the temporal constraint.

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.

For 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. It is obvious that the unsound edges w.r.t. *cur* should never appear in the induced subgraph of any supergroup of *cur*.

Example 19 Given the temporal graph in Fig. 3 (denoted as $\mathcal{G}$), $cur = \{v_1, v_2, v_3, v_4\}$ is a vertex group, and *maxspan* = 7 days. For vertex $v_5 \notin cur$, $(v_5, v_6, 68)$ is an unsound edge w.r.t. *cur*, because v_6 is an invalid neighbor of *cur*, and the clustered edge between v_5 and v_6 is unsound. Besides, 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 w.r.t. *cur*, because the clustered edges which these temporal edges belong to are all sound w.r.t. *cur*. For vertex v_7, the five temporal edges adjacent to it are all unsound, because the clustered edges which they belong to are unsound and violate the temporal constraint.

Lemma 2 *$\frac{a}{b} \le \frac{a+c}{b+d}$, if $ad \le bc$, where $a, c, d \ge 0$, $b > 0$.*

Based on Lemma 2, the following theorem is proposed.

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

Proof Given a vertex group *cur*, when a vertex $v \in V_q = V - cur$ is added into *cur*, the upper bound of the R value of $cur \cup \{v\}$ is $\frac{|\mathcal{E}_{cur}|+N^{(v)}_{sound}}{|\mathcal{E}^l_{cur}|+N^{(v)}_{unsound}}$. It corresponds to the condition that v's sound temporal edges are all adjacent to vertices in *cur*, and the unsound temporal edges are adjacent to vertices out of *cur*. By adding more vertices into *cur*, a supergroup with more vertices can be obtained, and its upper bound of R value is $\frac{|\mathcal{E}_{cur}|+\sum_{1\le j\le p} N^{(v_{n_j})}_{sound}}{|\mathcal{E}^l_{cur}|+\sum_{1\le j\le p} N^{(v_{n_j})}_{unsound}}$. Then, the problem is which vertices should be chosen to add into *cur* and form a supergroup, so that the upper bound of this supergroup's R value is the largest. The largest value (i.e., a new upper bound, denoted as R') is the upper bound of R of the supergroups of *cur*, i.e., $\widehat{R} = R'$, and R' is to be computed.

Please note that each value $\frac{N^{(v_{n_x})}_{sound}}{N^{(v_{n_x})}_{unsound}}$ corresponds to a vector in a planar, and the vector is $(N^{(v_{n_x})}_{unsound}, N^{(v_{n_x})}_{sound})$. This vector corresponds to vertex v_{n_x} and is denoted by $\boldsymbol{v}_{n_x}$. The vector corresponding to $\frac{|\mathcal{E}_{cur}|}{|\mathcal{E}^l_{cur}|}$ is denoted by $\boldsymbol{cur}$. Note that, the slope of $\boldsymbol{v}_{n_x}$ equals the value of $\frac{N^{(v_{n_x})}_{sound}}{N^{(v_{n_x})}_{unsound}}$ and is denoted by $\mathcal{S}(\boldsymbol{v}_{n_x})$. Then, the summation of $\boldsymbol{cur}$ and the vectors including $\boldsymbol{v}_{n_1}, \ldots, \boldsymbol{v}_{n_s}$ (the s vertices in V_q) can form many new vectors, and these vectors correspond to different supergroups of *cur*. For example, the vector $\boldsymbol{cur} + \boldsymbol{v}_{n_1}$ corresponds to the supergroup $cur \cup \{v_{n_1}\}$, and the slope of the vector has the same value as $\frac{|\mathcal{E}_{cur}|+N^{(v_{n_1})}_{sound}}{|\mathcal{E}^l_{cur}|+N^{(v_{n_1})}_{unsound}}$, which is the upper bound of the supergroup's R value. Thus, the problem of computing the $\widehat{R}$ value is transformed to obtaining the maximal slope that can be obtained by the summation of the vectors. In more detail, as the slope of a vector is the upper bound of a supergroup's R value, the maximal slope is the largest value of the upper bounds of the R values of the supergroups of *cur* and, therefore, is the value of $\widehat{R}$.

When $\frac{N^{(v_{n_1})}_{sound}}{N^{(v_{n_1})}_{unsound}} \ge \frac{|\mathcal{E}_{cur}|}{|\mathcal{E}^l_{cur}|}$, there is at least one vector $\boldsymbol{v}_{n_1}$ that has a larger slope than $\boldsymbol{cur}$. Therefore, according to Lemma 2, the largest slope that can be obtained is larger than $\mathcal{S}(\boldsymbol{cur})$. Assume that $\widehat{cur}$ is the supergroup of *cur* whose upper bound of R is the largest among the supergroups of *cur* (i.e., $\widehat{\boldsymbol{cur}}$ has the largest slope), and $\widehat{cur} = cur \cup V_m$, where $V_m \subseteq V - cur$ and V_m contains m vertices. The upper bound of $\widehat{cur}$'s R value equals $\mathcal{S}(\boldsymbol{cur} + \sum_{v_i \in V_m} \boldsymbol{v}_i)$. We prove that for two vectors $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ with $\mathcal{S}(\boldsymbol{v}_i) < \mathcal{S}(\boldsymbol{v}_j)$, it can never happen that $v_i \in \widehat{cur} \wedge v_j \notin \widehat{cur}$. Without loss of generality, suppose that $v_{n_{m+1}} \in V_m$, $v_{n_m} \notin V_m$, and $\mathcal{S}(\boldsymbol{v}_{n_{m+1}}) < \mathcal{S}(\boldsymbol{v}_{n_m})$. Then, if $\mathcal{S}(\boldsymbol{cur} + \sum_{v_i \in V_m} \boldsymbol{v}_i) < \mathcal{S}(\boldsymbol{v}_{n_m})$, we have $\mathcal{S}(\boldsymbol{cur} + \boldsymbol{v}_{n_m} + \sum_{v_i \in V_m} \boldsymbol{v}_i) > \mathcal{S}(\boldsymbol{cur} + \sum_{v_i \in V_m} \boldsymbol{v}_i)$. Otherwise, if $\mathcal{S}(\boldsymbol{cur} + \sum_{v_i \in V_m} \boldsymbol{v}_i) \ge \mathcal{S}(\boldsymbol{v}_{n_m}) > \mathcal{S}(\boldsymbol{v}_{n_{m+1}})$, we have $\mathcal{S}(\boldsymbol{cur} - \boldsymbol{v}_{n_{m+1}} + \sum_{v_i \in V_m} \boldsymbol{v}_i) > \mathcal{S}(\boldsymbol{cur} + \sum_{v_i \in V_m} \boldsymbol{v}_i)$. Therefore, $\mathcal{S}(\boldsymbol{cur} + \sum_{v_i \in V_m} \boldsymbol{v}_i)$ is not the maximal slope, and then $\widehat{cur}$ does not have the largest upper bound of R. It contrasts with the assumption. Hence, for $\widehat{cur} = cur \cup V_m$, which has the largest upper bound of R among the supergroups of *cur*, the m vertices in V_m should have the largest m slopes in $V - cur$. That is, $V_m = \{v_{n_1}, \ldots, v_{n_m}\}$. Then, the vertices with larger slopes should be added earlier. According to Lemma 2, when

the next vertex makes the slope smaller, the following vertices should not be added. Therefore, when $\frac{N_{sound}^{(v_{n_1})}}{N_{unsound}^{(v_{n_1})}} \geq \frac{|\mathcal{E}_{cur}|}{|\mathcal{E}_{cur}^l|}$, $\widehat{R} = \frac{|\mathcal{E}_{cur}| + \sum_{1\leq j\leq i} N_{sound}^{(v_{n_i})}}{|\mathcal{E}_{cur}^l| + \sum_{1\leq j\leq i} N_{unsound}^{(v_{n_i})}}$, where v_{n_i} is the first vertex in $v_{n_1}, \ldots, v_{n_s}$ satisfying $\frac{|\mathcal{E}_{cur}| + \sum_{1\leq j\leq i} N_{sound}^{(v_{n_i})}}{|\mathcal{E}_{cur}^l| + \sum_{1\leq j\leq i} N_{unsound}^{(v_{n_i})}} \geq \frac{N_{sound}^{(v_{n_i+1})}}{N_{unsound}^{(v_{n_i+1})}}$.

Next, when $\frac{N_{sound}^{(v_{n_1})}}{N_{unsound}^{(v_{n_1})}} < \frac{|\mathcal{E}_{cur}|}{|\mathcal{E}_{cur}^l|}$, there is no vector in $\{\boldsymbol{v}_{n_1}, \ldots, \boldsymbol{v}_{n_s}\}$ that has a larger slope than $\boldsymbol{cur}$. Therefore, the maximal slope that can be obtained is smaller than $\mathcal{S}(\boldsymbol{cur})$. At that time, the maximal slope is $\max(\frac{|\mathcal{E}_{cur}|+N_{sound}^{(v_{n_1})}}{|\mathcal{E}_{cur}^l|+N_{unsound}^{(v_{n_1})}}, \ldots, \frac{|\mathcal{E}_{cur}|+N_{sound}^{(v_{n_s})}}{|\mathcal{E}_{cur}^l|+N_{unsound}^{(v_{n_s})}})$, i.e., $\max(\mathcal{S}(\boldsymbol{cur}+\boldsymbol{v}_{n_1}), \ldots, \mathcal{S}(\boldsymbol{cur}+\boldsymbol{v}_{n_s}))$. Let $\mathcal{S}(\boldsymbol{cur}+\boldsymbol{v}_{n_p}) = \max(\mathcal{S}(\boldsymbol{cur}+\boldsymbol{v}_{n_1}), \ldots, \mathcal{S}(\boldsymbol{cur}+\boldsymbol{v}_{n_s}))$, which means that the vertex group $\widehat{cur} = cur \cup \{v_{n_p}\}$ has the largest upper bound of R among the supergroups of cur. Assume that there is a supergroup of cur, i.e., $cur' = cur \cup V_m = cur \cup \{v_{m_1}, \ldots, v_{m_k}\}\ (k \geq 2)$, having a larger upper bound of R than $\widehat{cur}$. It means that $\mathcal{S}(\boldsymbol{cur} + \sum_{v_{m_i}\in V_m} \boldsymbol{v}_{m_i}) > \mathcal{S}(\boldsymbol{cur}+\boldsymbol{v}_{n_p})$. If $\mathcal{S}(\boldsymbol{cur}+\boldsymbol{v}_{n_p}) \geq \max\left(\mathcal{S}(\boldsymbol{v}_{m_1}), \ldots, \mathcal{S}(\boldsymbol{v}_{m_k})\right)$, we have $\mathcal{S}(\boldsymbol{cur}+\boldsymbol{v}_{n_p}) \geq \mathcal{S}(\boldsymbol{cur} + \sum_{v_{m_i}\in V_m} \boldsymbol{v}_{m_i})$. Otherwise, if there exists $v_{m_{k_0}}$ satisfying that $\mathcal{S}(\boldsymbol{cur}+\boldsymbol{v}_{n_p}) < \mathcal{S}(\boldsymbol{v}_{m_{k_0}})$, we have $\mathcal{S}(\boldsymbol{cur}+\boldsymbol{v}_{n_p}) < \mathcal{S}(\boldsymbol{cur}+\boldsymbol{v}_{m_{k_0}})$. It contrasts the assumption. In conclusion, when $\frac{N_{sound}^{(v_{n_1})}}{N_{unsound}^{(v_{n_1})}} < \frac{|\mathcal{E}_{cur}|}{|\mathcal{E}_{cur}^l|}$, $\max(\mathcal{S}(\boldsymbol{cur} + \boldsymbol{v}_{n_1}), \ldots, \mathcal{S}(\boldsymbol{cur} + \boldsymbol{v}_{n_s}))$ is the maximal slope that can be obtained, i.e., $\widehat{R} = \max(\frac{|\mathcal{E}_{cur}|+N_{sound}^{(v_{n_1})}}{|\mathcal{E}_{cur}^l|+N_{unsound}^{(v_{n_1})}}, \ldots, \frac{|\mathcal{E}_{cur}|+N_{sound}^{(v_{n_s})}}{|\mathcal{E}_{cur}^l|+N_{unsound}^{(v_{n_s})}})$. □

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

Example 20 In Fig. 3 (i.e., $\mathcal{G} = (V, \mathcal{E})$), let $cur = \{v_1, v_2, v_3, v_4\}$. Then, $|\mathcal{E}_{cur}| = 8$, $|\mathcal{E}_{cur}^l| = 13$, and $maxspan = 7$. For the vertices in $V - cur$, $\frac{N_{sound}^{(v_5)}}{N_{unsound}^{(v_5)}} = \frac{5}{1}$, $\frac{N_{sound}^{(v_6)}}{N_{unsound}^{(v_6)}} = \frac{0}{6}$, $\frac{N_{sound}^{(v_7)}}{N_{unsound}^{(v_7)}} = \frac{0}{5}$, and $\frac{N_{sound}^{(v_8)}}{N_{unsound}^{(v_8)}} = \frac{1}{3}$. Therefore, $\widehat{R} = \frac{8+5}{13+1} = 0.93$, and $\widehat{\zeta}_{\text{inter}} = 1$.

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

Lemma 3 *Vertex groups with $\widehat{\zeta}_{inter} < \gamma$ are invalid.*

Algorithm 4: ComputeMaxTC

Input: $\mathcal{G} = (V, \mathcal{E})$: a temporal graph, *cur*: current vertex group, *ECL*: the ECL index on $\mathcal{G}$, γ: 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*

```
ζ̂_t = ComputeZetaT(G, cur) //Compute the value of ζ̂_t
if ζ̂_t < γ then
    return -1 // Lemma 1

validv, invalidv ← GetValidNeighbors(G, cur, maxspan, ECL)
ζ̂_inter = ComputeZetaInter(G, cur, ECL, maxspan, invalidv) //Compute the value of ζ̂_inter
if ζ̂_inter < γ then
    return -1 // Lemma 3

ζ̂_intra = ComputeZetaIntra(G, cur, ECL, maxspan, invalidv) //Compute the value of ζ̂_intra
if ζ̂_intra < γ then
    return -1 // Lemma 4

// Compute the upper bound of the T-cohesiveness
T̂_c = ζ̂_t * ζ̂_inter * ζ̂_intra return T̂_c
```

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

As stated in Eq. 5, the value of $\widehat{\zeta}_{\text{intra}}$ is determined by the upper bound of D (denoted as $\widehat{D}$), and the value of $\widehat{D}$ 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 $v_{n_1}, \ldots, v_{n_s}$ are all the vertices in V-cur, and satisfy $\frac{\min(k, N_{sound}^{c(v_{n_1})})}{k} \geq \cdots \geq \frac{\min(k, N_{sound}^{c(v_{n_s})})}{k}$. 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 $v_{n_1}, \ldots, v_{n_s}$ satisfying that $\frac{C + \sum_{1\leq j\leq i} \min(k, N_{sound}^{c(v_{n_j})})}{|cur|*k + i*k} \geq \frac{\min(k, N_{sound}^{c(v_{n_i+1})})}{k}$. Otherwise, we have $\widehat{D} = \max(\frac{C+\min(k, N_{sound}^{c(v_{n_1})})}{|cur|*k+k}, \ldots, \frac{C+\min(k, N_{sound}^{c(v_{n_s})})}{|cur|*k+k})$.*

Proof The proof is similar to that of Theorem 3 and omitted. □

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

Example 21 In Fig. 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 $\widehat{\zeta}_{\text{intra}} = 1$.

Moreover, let $cur = \{v_1, v_2, v_3, v_4, v_5\}$, and 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 $\widehat{\zeta}_{\text{intra}} = 0.69$.

Lemma 4 *Vertex groups with $\widehat{\zeta}_{intra} < \gamma$ are invalid.*

Algorithm 5: ComputeZetaT

Input: $\mathcal{G} = (V, \mathcal{E})$: a temporal graph, *cur*: current vertex group
Output: $\widehat{\zeta}_t$: the upper bound of ζ_t of the supergroups of *cur*

1 $maxt, mint \leftarrow$ the maximal and minimal timestamp in $I(cur)$
2 $T \leftarrow maxt - mint$
3 $\widehat{\zeta}_t = \frac{1}{1+log(\frac{e-1}{T_{0.5}-T_1}*(\max(T,T_1)-T_1)+1)}$
4 return $\widehat{\zeta}_t$

Algorithm 6: ComputeZetaInter

Input: $\mathcal{G} = (V, \mathcal{E})$: a temporal graph, *cur*: current vertex group, *ECL*: the ECL index on $\mathcal{G}$, *maxspan*: the maximal time span a combo can have, *invalidv*: invalid neighbors of vertices in *cur*
Output: $\widehat{\zeta}_{\text{inter}}$: the upper bound of ζ_{inter} of the supergroups of *cur*

1 $|\mathcal{E}_{cur}|, |\mathcal{E}^l_{cur}| \leftarrow$ the number of temporal edges in the induced subgraph and local structure of *cur*, respectively
2 $upper \leftarrow |\mathcal{E}_{cur}|$, $lower \leftarrow |\mathcal{E}^l_{cur}|$
3 *sound*, *unsound* $\leftarrow$ *List()*
4 *largerInter* $\leftarrow$ False
5 **foreach** $u \in V - cur - invalidv$ **do**
6 *outer, inner* $\leftarrow 0$
7 **foreach** $e^c \in ECL[u]$ **do**
8 **if** $\max(maxt, e^c.maxtime) - \min(mint, e^c.mintime) > maxspan$ **then**
9 *outer* += $e^c.num$
10 **else**
11 **if** $e^c.node1 \in invalidv$ *or* $e^c.node2 \in invalidv$ **then**
12 *outer* += $e^c.num$
13 **else**
14 *inner* += $e^c.num$
15 *sound.Push(inner)*
16 *unsound.Push(outer)*
17 **if** $\frac{inner}{outer} \geq \frac{|\mathcal{E}_{cur}|}{|\mathcal{E}^l_{cur}|}$ **then**
18 *largerInter = True*
19 sort *sound* and *unsound* in descending order of the value of $\frac{sound[i]}{unsound[i]}$
20 **if** *largerInter* **then**
21 **foreach** *i from 0 to* |*sound*| *- 1* **do**
22 **if** $\frac{upper}{lower} < \frac{sound[i]}{unsound[i]}$ **then**
23 *upper* += *sound[i]*
24 *lower* += *unsound[i]*
25 **else**
26 *Break*
27 $\widehat{\zeta}_{\text{inter}} = \frac{1}{1+log(\frac{e-1}{R_1-R_{0.5}}*(R_1-\min(\frac{upper}{lower},R_1))+1)}$
28 **else**
29 *maxrate = -1*
30 **foreach** *i from 0 to* |*sound*| *- 1* **do**
31 $maxrate = \max(maxrate, \frac{upper + sound[i]}{lower + unsound[i]})$
32 $\widehat{\zeta}_{\text{inter}} = \frac{1}{1+log(\frac{e-1}{R_1-R_{0.5}}*(R_1-\min(maxrate,R_1))+1)}$
33 return $\widehat{\zeta}_{\text{inter}}$

Algorithm 7: ComputeZetaIntra

Input: $\mathcal{G} = (V, \mathcal{E})$: a temporal graph, *cur*: current vertex group, *ECL*: the ECL index on $\mathcal{G}$, *maxspan*: the maximal time span a combo can have, *invalidv*: invalid neighbors of vertices in *cur*
Output: $\widehat{\zeta}_{\text{intra}}$: the upper bound of ζ_{intra} of the supergroups of *cur*

1 $C \leftarrow \sum_{v_i \in cur} \min(k, N^{c(v_i)}_{valid})$
2 $nodenum \leftarrow |cur|$
3 $stdcore \leftarrow \frac{C}{|cur|*k}$
4 *corelist* $\leftarrow$ *List()*
5 *largerIntra* $\leftarrow$ False
6 **foreach** $u \in V - cur - invalidv$ **do**
7 *corenum* $\leftarrow 0$
8 **foreach** $e^c \in ECL[u]$ **do**
9 **if** $\max(maxt, e^c.maxtime) - \min(mint, e^c.mintime) \leq maxspan$ **then**
10 **if** $e^c.node1 \notin invalidv$ *and* $e^c.node2 \notin invalidv$ **then**
11 *corenum* += 1
12 **if** $\frac{\min(k,corenum)}{k} \geq stdcore$ **then**
13 *largerIntra* $\leftarrow$ True
14 *corelist.Push*($\min(k, corenum)$)
15 sort *corelist* in descending order
16 **if** *largerIntra* **then**
17 **foreach** *i from 0 to* |*corelist*| *- 1* **do**
18 **if** $\frac{C}{nodenum*k} < \frac{corelist[i]}{k}$ **then**
19 C += *corelist[i]*
20 *nodenum* += *1*
21 **else**
22 *Break*
23 $\widehat{\zeta}_{\text{intra}} = \frac{1}{1+log(\frac{e-1}{D_1-D_{0.5}}*(D_1-\min(\frac{C}{nodenum*k},D_1))+1)}$
24 **else**
25 *maxrate = -1*
26 **foreach** *i from 0 to* |*corelist*| *- 1* **do**
27 $maxrate = \max(maxrate, \frac{C + corelist[i]}{(nodenum + 1)*k})$
28 $\widehat{\zeta}_{\text{intra}} = \frac{1}{1+log(\frac{e-1}{D_1-D_{0.5}}*(D_1-\min(maxrate,D_1))+1)}$
29 return $\widehat{\zeta}_{\text{intra}}$

5.4 Upper bound of $\mathbb{T}$-cohesiveness

With $\widehat{\zeta}_t$, $\widehat{\zeta}_{\text{inter}}$, and $\widehat{\zeta}_{\text{intra}}$ analyzed above, Algorithm 4 named *ComputeMaxTC* is proposed to compute the $\mathbb{T}$-cohesiveness upper bound of the supergroups of a given vertex group.

Specifically, the values of $\widehat{\zeta}_t$, $\widehat{\zeta}_{\text{inter}}$, and $\widehat{\zeta}_{\text{intra}}$ are computed at Line 1, Line 6, and Line 10, respectively. Line 5 invokes *GetValidNeighbors* to obtain *invalidv*, which consists of all the invalid neighbors of vertices in *cur*.

The algorithms to compute the values of $\widehat{\zeta}_t$, $\widehat{\zeta}_{\text{inter}}$, and $\widehat{\zeta}_{\text{intra}}$ are shown in Algorithms 5, 6, and 7, respectively. In Algorithm 5, the time span of $I(cur)$ is computed at Line 1 and Line 2. Then, the upper bound of the TC score of the supergroups of *cur* is obtained (Line 3).

In Algorithm 6, the numbers of temporal edges in $I(cur)$ and in the local structure of *cur* are first computed at Line 1 (i.e., $|\mathcal{E}_{cur}|$ and $|\mathcal{E}^l_{cur}|$), and the ratio of the intra-edges of $I(cur)$ is $\frac{|\mathcal{E}_{cur}|}{|\mathcal{E}^l_{cur}|}$. Then, all the vertices that can be added into *cur* are traversed in Lines 5–18. The vertices in *invalidv* are dropped, because for any vertex $u \in invalidv$, $I(cur \cup \{u\})$ has a time span longer than *maxspan*. For each vertex u traversed, each clustered edge e^c adjacent to u is processed. If e^c makes the time span of $I(cur \cup \{u\})$ exceed *maxspan*, temporal edges in e^c are unsound, because e^c violates the temporal constraint and is unsound w.r.t. *cur* (Lines 8–9). Moreover, if e^c connects u and a vertex in *invalidv*, temporal edges in e^c are also unsound, because e^c violates the vertex constraint (Lines 11–12). Otherwise, e^c is sound. After all the clustered edges adjacent to u are processed, the numbers of sound temporal edges and unsound temporal edges adjacent to u are recorded in *inner* and *outer*, respectively. Then, these *inner* and *outer* are pushed into two lists, i.e., *sound* and *unsound*, respectively. Lines 17–18 check whether there exists a vertex $u \in V - cur - validv$ satisfying that $\frac{N^{(u)}_{sound}}{N^{(u)}_{unsound}} \geq \frac{|\mathcal{E}_{cur}|}{|\mathcal{E}^l_{cur}|}$.

Lines 19–32 compute the values of $\widehat{R}$ and $\widehat{\zeta}_{\text{inter}}$ according to Theorem 3.

Algorithm 7 computes the value of $\widehat{\zeta}_{\text{intra}}$. In detail, Lines 1–3 compute the upper bound of the density of the vertices in *cur*, which is $stdcore = \frac{C}{|cur|*k}$. Then, vertices that can be added into *cur* are traversed in Lines 6–14. For each clustered edge e^c adjacent to the processed vertex u, it is checked whether e^c is sound w.r.t. *cur* (Lines 9–11), and every time a sound clustered edge adjacent to u is found, the value of *corenum* increases by one. Please note that the value of *corenum* is the maximal degree that u can have in a supergroup of *cur*. After all the clustered edges adjacent to u are traversed, *corenum* is pushed into *corelist* (Line 14). Specifically, Lines 12–13 check whether there exists a vertex $u \in V - cur - validv$ satisfying $\frac{\min(k, N_{sound}^{c(u)})}{k} \geq \frac{C}{|cur|*k}$. Then, Lines 15–28 compute the values of $\widehat{D}$ and $\widehat{\zeta}_{\text{intra}}$ according to Theorem 4.

The time complexities of Algorithms 5, 6, and 7 are $O(|E^P|)$, $O(|E^P|+\Theta(|V|))$, and $O(|E^P|+\Theta(|V|))$, respectively, where $|E^P|$ is the number of edges in the projected graph of $\mathcal{G}$. Moreover, the time complexity of Algorithm 4 is $O(|E^P| + \Theta(|V|))$. Please note that $\Theta(|V|)$ is the time complexity of the sorting algorithm.

Example 22 Given Fig. 3, $cur = \{v_1, v_2, v_3, v_4, v_5\}$ is obtained at Line 9 of Algorithm 3. When *ComputeMaxTC* is invoked at Line 18, we obtain that $\widehat{\zeta}_t = 0.69$, $\widehat{\zeta}_{\text{inter}} = 0.68$, $\widehat{\zeta}_{\text{intra}} = 0.69$, and $\widehat{\mathbb{T}}_c = 0.32 < \gamma = 0.5$. Thus, supergroups of *cur* cannot form a combo, and *cur* is not inserted into Q.

Theorem 5 *Given temporal graph $\mathcal{G} = (V, \mathcal{E})$, and a vertex group $cur \subset V$, $\widehat{\zeta}_t$ is the supremum of the ζ_t value of the supergroups of cur, when there is a valid neighbor of vertices in cur (denoted as u) satisfying that the time span of $I(cur \cup \{u\})$ is smaller than T_1 or is equal to that of $I(cur)$.*

Proof According to Theorem 2, $\widehat{\zeta}_t$ is the upper bound of the ζ_t values of the supergroups of *cur*. Then, it should be proved that the value of $\widehat{\zeta}_t$ can be obtained. When there is a valid neighbor of vertices in *cur* (denoted as u) satisfying that the time span of $I(cur \cup \{u\})$ is smaller than T_1 or is equal to that of $I(cur)$, the ζ_t value of $I(cur \cup \{u\})$ equals $\zeta_t(I(cur)) = \widehat{\zeta}_t$. Therefore, $\widehat{\zeta}_t$ is achieved and is the supremum on that condition. □

Theorem 6 *Given temporal graph $\mathcal{G} = (V, \mathcal{E})$, and a vertex group $cur \subset V$, suppose $v_{n_1}, \ldots, v_{n_s}$ are all the vertices in $V - cur$, and satisfy $\frac{N_{sound}^{(v_{n_1})}}{N_{unsound}^{(v_{n_1})}} \geq \cdots \geq \frac{N_{sound}^{(v_{n_s})}}{N_{unsound}^{(v_{n_s})}}$. Let v_{n_i} be the first vertex in $v_{n_1}, \ldots, v_{n_s}$ satisfying $\frac{|\mathcal{E}_{cur}| + \sum_{1 \leq j \leq i} N_{sound}^{(v_{n_i})}}{|\mathcal{E}_{cur}^l| + \sum_{1 \leq j \leq i} N_{unsound}^{(v_{n_i})}} \geq \frac{N_{sound}^{(v_{n_i+1})}}{N_{unsound}^{(v_{n_i+1})}}$. Then, if $\frac{N_{sound}^{(v_{n_1})}}{N_{unsound}^{(v_{n_1})}} \geq \frac{|\mathcal{E}_{cur}|}{|\mathcal{E}_{cur}^l|}$, $\widehat{\zeta}_{inter}$ is the supremum of the ζ_{inter} value of the supergroups of cur, when for the vertices $v_{n_1}, \ldots, v_{n_i}$, their sound temporal edges are all adjacent to vertices in cur, while their unsound temporal edges are all adjacent to vertices out of $cur \cup \{v_{n_1}, \ldots, v_{n_i}\}$. Otherwise, $\widehat{\zeta}_{inter}$ is the supremum of the $\widehat{\zeta}_{inter}$ value of the supergroups of cur, when for the vertex v_{n_p} satisfying $\frac{|\mathcal{E}_{cur}| + N_{sound}^{(v_{n_p})}}{|\mathcal{E}_{cur}^l| + N_{unsound}^{(v_{n_p})}} = \max(\frac{|\mathcal{E}_{cur}| + N_{sound}^{(v_{n_1})}}{|\mathcal{E}_{cur}^l| + N_{unsound}^{(v_{n_1})}}, \ldots, \frac{|\mathcal{E}_{cur}| + N_{sound}^{(v_{n_s})}}{|\mathcal{E}_{cur}^l| + N_{unsound}^{(v_{n_s})}})$, v_{n_p}'s sound temporal edges are all adjacent to the vertices in cur, while v_{n_p}'s unsound temporal edges are all adjacent to vertices out of $cur \cup \{v_{n_p}\}$.*

Proof According to Theorem 3, $\widehat{\zeta}_{\text{inter}}$ is the upper bound of the ζ_{inter} values of the supergroups of *cur*. Then, it should be proved that the value of $\widehat{\zeta}_{\text{inter}}$ can be obtained. Firstly, if $\frac{N_{sound}^{(v_{n_1})}}{N_{unsound}^{(v_{n_1})}} \geq \frac{|\mathcal{E}_{cur}|}{|\mathcal{E}_{cur}^l|}$, on the condition that for the vertices $v_{n_1}, \ldots, v_{n_i}$, their sound temporal edges are all adjacent to vertices in *cur*, while their unsound temporal edges are all adjacent to vertices out of $cur \cup \{v_{n_1}, \ldots, v_{n_i}\}$, the R value (rather than an upper bound) of $I(cur \cup \{v_{n_1}, \ldots, v_{n_i}\})$ is $\frac{|\mathcal{E}_{cur}| + \sum_{1 \leq j \leq i} N_{sound}^{(v_{n_j})}}{|\mathcal{E}_{cur}^l| + \sum_{1 \leq j \leq i} N_{unsound}^{(v_{n_j})}}$. Since it is the same as the value of $\widehat{R}$, $\widehat{\zeta}_{\text{inter}}$ is obtained. Thus, $\widehat{\zeta}_{\text{inter}}$ is a supremum on that condition.

Secondly, if $\frac{N_{sound}^{(v_{n_1})}}{N_{unsound}^{(v_{n_1})}} < \frac{|\mathcal{E}_{cur}|}{|\mathcal{E}_{cur}^l|}$, on the condition that v_p's sound temporal edges are all adjacent to the vertices in *cur*, while v_{n_p}'s unsound temporal edges are all adjacent to vertices out of $cur \cup \{v_{n_p}\}$, the R value (rather than an upper bound) of $I(cur \cup \{v_{n_p}\})$ is $\frac{|\mathcal{E}_{cur}| + N_{sound}^{(v_{n_p})}}{|\mathcal{E}_{cur}^l| + N_{unsound}^{(v_{n_p})}}$, which equals the value of $\widehat{R}$, and $\widehat{\zeta}_{\text{inter}}$ is obtained. Thus, $\widehat{\zeta}_{\text{inter}}$ is a supremum on that condition. In conclusion, Theorem 6 is correct. □

Theorem 7 *Given $\mathcal{G} = (V, \mathcal{E})$, a vertex group $cur \subset V$, and core number k, suppose $v_{n_1}, \ldots, v_{n_s}$ are all the vertices in $V - cur$, and satisfy $\frac{\min(k, N_{sound}^{c(v_{n_1})})}{k} \geq \cdots \geq \frac{\min(k, N_{sound}^{c(v_{n_s})})}{k}$. Let $C = \sum_{v \in cur} \min(k, N_{sound}^{c(v)})$, and let v_{n_i} be the first vertex in $v_{n_1}, \ldots, v_{n_s}$ satisfying that $\frac{C + \sum_{1 \leq j \leq i} \min(k, N_{sound}^{c(v_{n_j})})}{|cur|*k + i*k} \geq \frac{\min(k, N_{sound}^{c(v_{n_i+1})})}{k}$. Then, if $\frac{\min(k, N_{sound}^{c(v_{n_1})})}{k} \geq \frac{C}{|cur|*k}$, $\widehat{\zeta}_{intra}$ is the supremum of the ζ_{intra} value of the supergroups of cur, when for each vertex u in $cur \cup \{v_{n_1}, \ldots, v_{n_i}\}$, u has at least $\min(k, N_{sound}^{c(u)})$ sound clustered edges adjacent to the vertices in $cur \cup \{v_{n_1}, \ldots, v_{n_i}\}$, and u's unsound temporal edges are all adjacent to the vertices out of $cur \cup \{v_{n_1}, \ldots, v_{n_i}\}$. Otherwise, $\widehat{\zeta}_{intra}$ is the supremum of the ζ_{intra} value of the supergroups of cur, when for vertex v_p satisfying $\frac{C + \min(k, N_{sound}^{c(v_{n_p})})}{|cur|*k + k} = \max(\frac{C + \min(k, N_{sound}^{c(v_{n_1})})}{|cur|*k + k}, \ldots, \frac{C + \min(k, N_{sound}^{c(v_{n_s})})}{|cur|*k + k})$, it is satisfied that*

for each vertex u in $cur \cup \{v_{n_p}\}$, u has at least $\min(k, N^{c(u)}_{sound})$ sound clustered edges adjacent to vertices in $cur \cup \{v_{n_p}\}$, and u's unsound temporal edges are all adjacent to vertices out of $cur \cup \{v_{n_p}\}$.

Proof The proof is similar to that of Theorem 6 and omitted. □

6 Experiments

In this section, the datasets and parameter configurations used in the experiments are first introduced. Then, the results of the performance evaluation, comparison experiment, memory cost evaluation, ablation study, scalability evaluation, parameter sensitivity, and case study are reported and analyzed in Sects. 6.2–6.8.

6.1 Experimental setting

Thirteen real-world datasets and four synthetic datasets are used in our experiments. The real-world datasets are extracted from ***CollegeMsg*** [35], ***Email*** [36], ***Contact*** [40], ***Math*** [36] 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. ***Math*** is from Mathoverflow, where the temporal edges represent the communication among the users. ***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 Sect. 3.7.

For these real-world datasets, 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 $[t_s, t_s + 86400\ \text{s}]$ into Col-1d. The other 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 are assigned to the edges. The timestamps are within the period of 1 day, i.e., the timestamps are in the range of $[1, 86400]$, and the unit of timestamp is second.

Table 3 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. 'd' and 'y' are the short forms of 'day' and 'year,' respectively. For

Table 3 Statistics of datasets

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert\mathcal{E}\rvert$	Time span	Dataset	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert\mathcal{E}\rvert$	Time span	Dataset	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert\mathcal{E}\rvert$	Time span
Col-1d	1899	102	314	1d	Cont-1d	10,972	3003	35,994	1d	Dblp-4y	364,605	1,032,437	2,700,430	4y
Col-7d	1899	326	1838	7d	Cont-7d	10,972	7170	110,134	7d	Syn_{1000}	1000	1270	1295	1d
Col-30d	1899	1069	6646	30d	Cont-30d	10,972	18,164	322,268	30d	Syn_{10000}	10,000	9345	10,461	1d
Email-1d	986	592	1904	1d	Math-1d	24,818	179	464	1d	Syn_{100000}	100,000	627,798	838,121	1d
Email-7d	986	1258	6488	7d	Math-7d	24,818	1084	3160	7d	Syn_{1M}	1,000,000	7,789,875	11,442,996	1d
Email-30d	986	1938	13,942	30d	Math-30d	24,818	2195	7612	30d					

Table 4 Configurations of parameters

CF. ID	$(T_1, T_{0.5})$	$(R_1, R_{0.5})$	$(k, D_1, D_{0.5})$	CF. ID	$(T_1, T_{0.5})$	$(R_1, R_{0.5})$	$(k, D_1, D_{0.5})$	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 (*)	(4d, 7d)	(0.9, 0.4)	(2, 1, 0.5)	29 (*)	(14d, 28d)	(0.9, 0.3)	(2, 1, 0.5)
2	(**0.4d**, 1d)	(0.9, 0.4)	(2, 1, 0.5)	16	(**3.5d**, 7d)	(0.9, 0.4)	(2, 1, 0.5)	30	(**13d**, 28d)	(0.9, 0.3)	(2, 1, 0.5)
3	(0.5d, **0.9d**)	(0.9, 0.4)	(2, 1, 0.5)	17	(4d, **6d**)	(0.9, 0.4)	(2, 1, 0.5)	31	(14d, **27d**)	(0.9, 0.3)	(2, 1, 0.5)
4	(0.5d, 1d)	(**0.95**, 0.4)	(2, 1, 0.5)	18	(4d, 7d)	(**0.95**, 0.4)	(2, 1, 0.5)	32	(14d, 28d)	(**0.95**, 0.3)	(2, 1, 0.5)
5	(0.5d, 1d)	(0.9, **0.5**)	(2, 1, 0.5)	19	(4d, 7d)	(0.9, **0.5**)	(2, 1, 0.5)	33	(14d, 28d)	(0.9, **0.35**)	(2, 1, 0.5)
6	(0.5d, 1d)	(0.9, 0.4)	(**3**, 1, **2/3**)	20	(4d, 7d)	(0.9, 0.4)	(**3**, 1, **2/3**)	34	(14d, 28d)	(0.9, 0.3)	(**3**, 1, **2/3**)
7	(0.5d, 1d)	(0.9, 0.4)	(2, **0.95**, 0.5)	21	(4d, 7d)	(0.9, 0.4)	(2, **0.95**, 0.5)	35	(14d, 28d)	(0.9, 0.3)	(2, **0.95**, 0.5)
8 (*)	(0.5d, 1d)	(0.9, 0.3)	(2, 1, 0.5)	22 (*)	(5d, 7d)	(0.9, 0.3)	(2, 1, 0.5)	36 (*)	(0y, 2y)	(0.9, 0.3)	(2, 1, 0.5)
9	(**0.4d**, 1d)	(0.9, 0.3)	(2, 1, 0.5)	23	(**4d**, 7d)	(0.9, 0.3)	(2, 1, 0.5)	37	(**1y**, 2y)	(0.9, 0.3)	(2, 1, 0.5)
10	(0.5d, **0.9d**)	(0.9, 0.3)	(2, 1, 0.5)	24	(5d, **6d**)	(0.9, 0.3)	(2, 1, 0.5)	38	(0y, **1y**)	(0.9, 0.3)	(2, 1, 0.5)
11	(0.5d, 1d)	(**0.95**, 0.3)	(2, 1, 0.5)	25	(5d, 7d)	(**0.95**, 0.3)	(2, 1, 0.5)	39	(0y, 2y)	(**0.95**, 0.3)	(2, 1, 0.5)
12	(0.5d, 1d)	(0.9, **0.35**)	(2, 1, 0.5)	26	(5d, 7d)	(0.9, **0.35**)	(2, 1, 0.5)	40	(0y, 2y)	(0.9, **0.35**)	(2, 1, 0.5)
13	(0.5d, 1d)	(0.9, 0.3)	(**3**, 1, **2/3**)	27	(5d, 7d)	(0.9, 0.3)	(**3**, 1, **2/3**)	41	(0y, 2y)	(0.9, 0.3)	(**3**, 1, **2/3**)
14	(0.5d, 1d)	(0.9, 0.3)	(2, **0.95**, 0.5)	28	(5d, 7d)	(0.9, 0.3)	(2, **0.95**, 0.5)	42	(0y, 2y)	(0.9, 0.3)	(2, **0.95**, 0.5)

Dblp-4y, the unit of timestamp is 1 year, while for each of the other datasets, the unit of timestamp is one second.

Moreover, the parameter configurations (abbr. CF.) applied in the experiments are shown in Table 4. Every seven configurations compose a group, and the first configuration in a group is the base of the group and marked as "(*)". For example, the first seven configurations form a group, and CF. 1 is the base. Specifically, CF. 2–CF. 7 are obtained by changing the value 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, 256GB RAM, and Windows Server 2008 operating system.

6.2 Performance evaluation

In this subsection, the performances of the proposed two time-topology analysis methods are evaluated on all the thirteen real-world datasets, and the parameter configurations applied on the datasets are shown in Table 5.

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. For each obtained combo $\mathcal{G}_s$, its time span, ratio of intra-edges, and density (denoted as T, R, and D, resp.) 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 T, R, and D values 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 baselines are the k-core community search algorithm [8] (denoted as KCA). Given a temporal graph $\mathcal{G}$ and a query vertex u, KCA finds the maximal k-core community that contains u, and the T, R, and D of the community are recorded. Specifically, KCA_3 and KCA_4 that search for 3-core communities and 4-core communities, respectively, are used in the experiments. The third baseline is Dynamo [52], which is proposed for dynamic community detection based on the widely used Louvain algorithm [3]. Because 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. Given a temporal graph and a query vertex u, Dynamo detects the community structure in the temporal graph, and 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 Fig. 8. Please 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 Fig. 8d, h, i, j, l, m, because when $\text{CS}_{0.6}$ is

Table 5 Configurations in performance experiments

Dataset	CF. ID	Dataset	CF. ID
Col-1d, Email-1d, Cont-1d	CF. 1	Math-7d	CF. 22
Math-1d	CF. 8	Col-30d, Email-30d, Cont-30d, Math-30d	CF. 29
Col-7d, Email-7d, Cont-7d	CF. 15	Dblp-4y	CF. 36

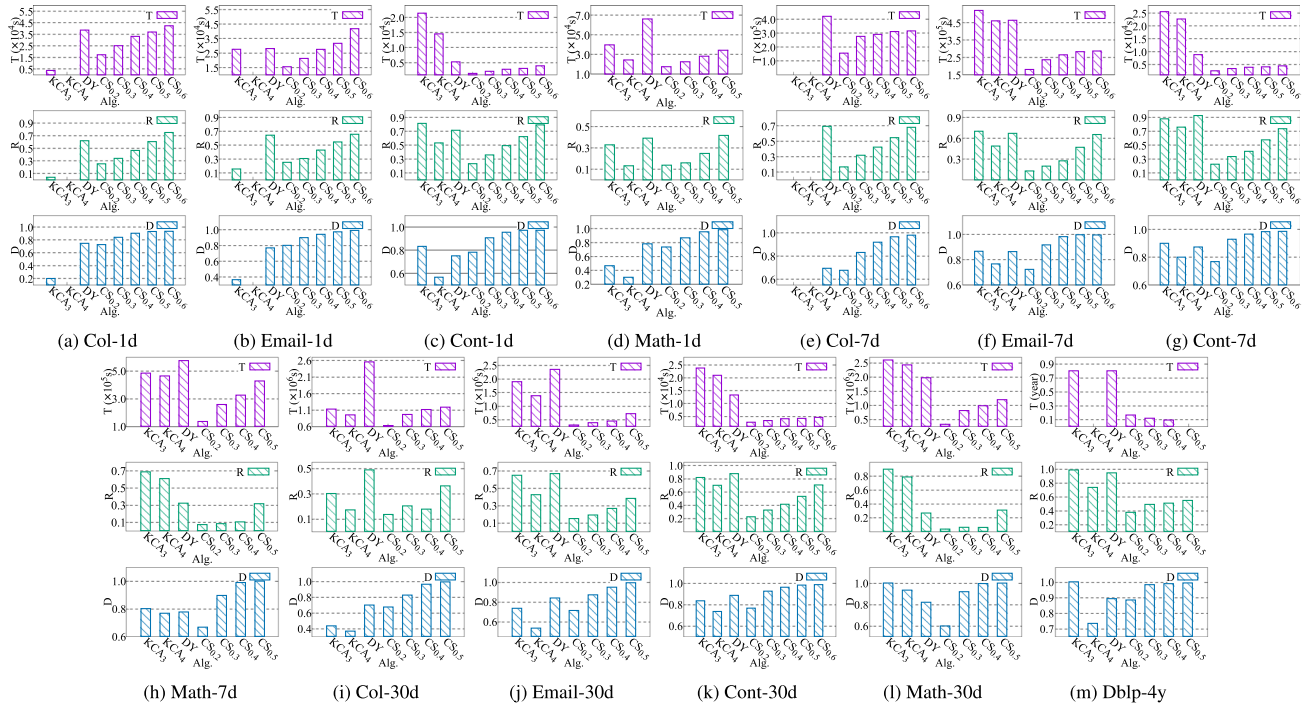


Fig. 8 The results of the performance experiments. T, R, and D represent the average time spans, ratios of the intra-edges, and densities of the obtained combos. Their values are reported with purple, green, and blue histograms, respectively (colour figure online)

performed on these datasets, 30 combos cannot be obtained in 30 min for most query vertices, and the experiments are stopped early.

6.2.1 Performance of $\mathbb{T}$-cohesiveness evolution tracking

Because the performance of $\mathbb{T}$-cohesiveness evolution tracking is determined by whether $\mathbb{T}$-cohesiveness can well evaluate the cohesiveness of a temporal subgraph, we confirm the superiority of $\mathbb{T}$-cohesiveness evolution tracking by showing that $\mathbb{T}$-cohesiveness can correctly measure the cohesiveness of temporal graphs. As shown in Fig. 8, with the increase in γ, the T, R, and D values of the combos obtained by *CS* always increase, which indicates that a temporal subgraph with a larger value of $\mathbb{T}$-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 increase in T may result from 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 time spans become larger with the increase in 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 (since they have large values of R and D). The experiment on Dblp-4y is a special case where the value of T decreases when γ increases. The reason may be that CF. 36 is used to conduct experiments on Dblp-4y, and T_1 is set to 0y. It indicates that the TC score will decrease to be much smaller than 1 when T increases. Although a temporal subgraph of Dblp-4y with a larger T may have large values of R and D, the decreasing TC score can make the $\mathbb{T}$-cohesiveness of the subgraph smaller than γ. Therefore, when experiments are conducted on Dblp-4y with CF. 36, the values of T decrease with the increase in γ to acquire large values of $\mathbb{T}$-cohesiveness. According to the above analysis, a temporal subgraph with a larger value of $\mathbb{T}$-cohesiveness is always more cohesive.

Moreover, it is noted that combos with large values of $\mathbb{T}$-cohesiveness are cohesive in both the time and topology dimensions. In terms of the temporal cohesiveness, on Col-1d, Email-1d, Cont-1d, and Math-1d, the obtained combos always have Ts shorter than half a day. It indicates that all the connections among the vertices in the obtained combos happen within 12 h. For the datasets with a time span of 7 days, the Ts of *CS* are always shorter than 5 days. For the datasets with a time span of 30 days, the average time spans of the obtained combos are always shorter than 14 days. Specifically, the average time spans of the obtained combos on Col-30d and Math-30d are about 13 days, while those of the obtained combos on Email-30d and Cont-30d are both shorter than 9 days. The TC scores of these obtained combos are all 1, and the experimental results confirm that the obtained combos are temporally cohesive. In terms of the inter-topological cohesiveness, for Col-1d, Email-1d, Cont-1d, Col-7d, Email-7d, Cont-7d, and Cont-30d, when $\gamma = 0.6$, the values of R are always larger than 0.65, which means that more than 65 percent of the temporal edges adjacent to the vertices in a combo are within the combo. For the other datasets, when $\gamma = 0.5$, it is also shown that more than 30 percent of temporal edges adjacent to the vertices in a combo are within the combo. In terms of the intra-topological cohesiveness, when $\gamma \geq 0.5$, the value of D is usually larger than 0.93 for all the datasets. It means that given an obtained combo with n vertices, fewer than $\frac{(1-0.93)*n*k}{2} = 0.07n$ (saying $k = 2$) edges are necessary to make the subgraph become a 2-core, which indicates that the vertices in the obtained combos are densely connected.

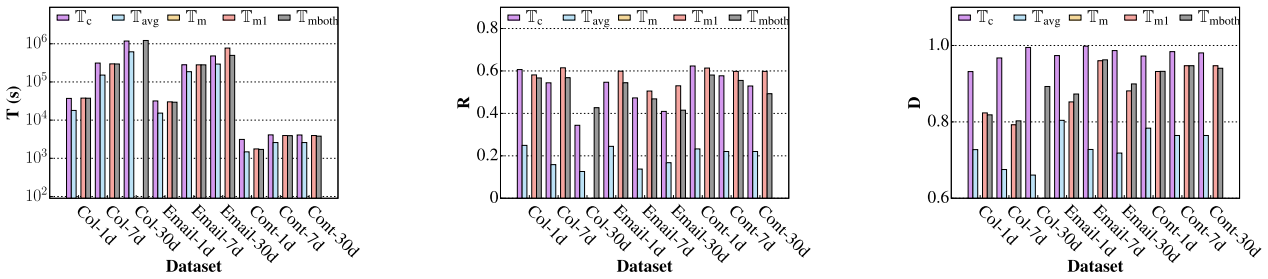
Therefore, the above analysis shows that $\mathbb{T}$-cohesiveness can well evaluate the cohesiveness of temporal subgraphs. Besides, because subgraphs with quite long time spans are considered cohesive by KCAs and Dynamo (e.g., on Email-7d and Email-30d, and Math-30d), it is demonstrated that KCAs and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

6.2.2 Performance of combo searching

The experimental results also confirm the superiority of the analysis of combo searching. Specifically, the combos obtained by *CS* are cohesive in both the time and topology dimensions as stated in Sect. 6.2.1. Also, the experimental results suggest that *CS* has better performance than the baseline methods when the three terms are considered simultaneously. Please note that on Math-1d, Math-7d, Col-30d, Email-30d, Math-30d, and Dblp-4y, the baselines are compared with $CS_{0.5}$. When the other datasets are used, the baselines are compared with $CS_{0.6}$. The combos found by Dynamo always have larger Ts and smaller Ds than those obtained by *CS*. For example, on Cont-7d, the average T of *CS* is about half that of Dynamo, and the average D of our method is much larger than that of Dynamo. A special case is the Email-1d dataset, where the average T of *CS* is longer than that of Dynamo. It may be because the combos obtained by *CS* contain more vertices and temporal edges to achieve a large value of R, and obtain a much larger value of D than Dynamo. It indicates that the combos found by *CS* on the Email-1d dataset focus on the topological cohesiveness and are much more cohesive than those found by Dynamo in the topology dimension. In terms of the value of R, *CS* often has similar R as Dynamo (e.g., on Col-1d and Email-7d). However, because Dynamo focuses on maximizing the value of R in the process of combo searching, it sometimes has larger R than *CS*, e.g., on Col-30d and Email-30d. Please note that 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 better from the perspectives of temporal cohesiveness and intra-topological cohesiveness. Therefore, *CS* has better performance than Dynamo.

Meanwhile, *CS* also has better performance than 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 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. However, the baselines cannot adapt to the varied conditions.



(a) Ts of the obtained cohesive subgraphs. (b) Rs of the obtained cohesive subgraphs. (c) Ds of the obtained cohesive subgraphs.

Fig. 9 Ts, Rs, and Ds of the combos obtained under different definitions of $\mathbb{T}$-cohesiveness

6.3 Comparison with different definitions of $\mathbb{T}$-cohesiveness

In this subsection, the other possible definitions of $\mathbb{T}$-cohesiveness shown in Table 2 and detailed in Sect. 3.6 are compared with our definition in Eq. 1 to show the superiority of our definition. Specifically, experiments are conducted on real-world datasets, and the configurations are the same as those used in Sect. 6.2. Besides, γ is set to 0.5.

The experimental results are shown in Fig. 9. Please note that the T, R, and D values when $\mathbb{T}_m$ is used are all zeros, because no combos can be found in 30 min for all the query vertices. Besides, the T, R, and D values when $\mathbb{T}_{m1}$ is used on Col-30d are also zeros, because for almost all query vertices (i.e., 29/30), 30 combos cannot be found in 30 min. To make a fair comparison, the same query vertices are used when different definitions of $\mathbb{T}$-cohesiveness are applied. Moreover, if 30 combos cannot be found for a query vertex under one definition of $\mathbb{T}$-cohesiveness (e.g., $\mathbb{T}_{m1}$), then this query vertex's combos obtained under all the definitions of $\mathbb{T}$-cohesiveness are omitted.

As shown in Fig. 9, the combos obtained with $\mathbb{T}_c$ are always more cohesive than the combos obtained with the other definitions, which confirms the efficiency of $\mathbb{T}_c$. In detail, although the combos obtained with $\mathbb{T}_{avg}$ have shorter time spans than those obtained with $\mathbb{T}_c$, these combos have quite small R and D values and are not cohesive. The possible reason is that the subgraphs cohesive only in the time dimension are considered to be combos according to $\mathbb{T}_{avg}$. Moreover, since no combos can be obtained with $\mathbb{T}_m$, $\mathbb{T}_c$ is better than $\mathbb{T}_m$. Besides, the combos obtained with $\mathbb{T}_c$ always have similar and competitive T and R values as those obtained with $\mathbb{T}_{m1}$ and $\mathbb{T}_{mboth}$. Meanwhile, the combos obtained with $\mathbb{T}_c$ always have much larger D values than those obtained with the other definitions, which illustrates that the combos obtained with $\mathbb{T}_c$ are more cohesive. Please note that the combos obtained with $\mathbb{T}_c$ sometimes have larger Ts than those obtained with $\mathbb{T}_{m1}$ and $\mathbb{T}_{mboth}$ (e.g., on Cont-1d). However, the time spans of these combos are all shorter than T_1 and are cohesive in the time dimension. Furthermore, on Email-30d, the combos obtained with $\mathbb{T}_{m1}$ and $\mathbb{T}_{mboth}$ have larger Rs than those obtained with $\mathbb{T}_c$. However, these combos have longer time spans and much smaller D values than those obtained with $\mathbb{T}_c$, and the combos obtained with $\mathbb{T}_c$ are more cohesive. Therefore, on the whole, $\mathbb{T}_c$ has better performance than the other definitions.

6.4 Memory cost evaluation

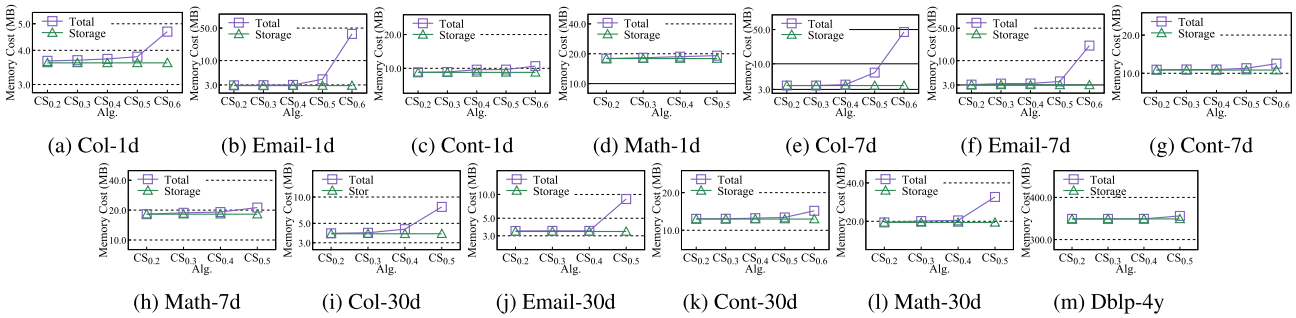
The memory costs (abbr. *mc*s) of $\mathbb{T}$-cohesiveness evolution tracking and combo searching are evaluated on the real-world datasets. The configurations applied are the same as those used in Sect. 6.2. Specifically, 30 query vertices are tested and their average *mc*s on each dataset are reported.

For $\mathbb{T}$-cohesiveness evolution tracking, both the width and the step length of the time window are set to 1 day, and the results are reported in Table 6. It is shown that with the increase in the group size, the *mc* increases slightly. Besides, with the increase in the temporal edge number (e.g., Cont-1d → Cont-7d → Cont-30d), the *mc* becomes larger. The results are consistent with the space complexity presented in Sect. 4.1.

For combo searching, the results are shown in Fig. 10. The purple lines represent the total memory costs, and the green lines represent the memory costs of storing the datasets and indexes. Therefore, the difference between the purple line and the green line is the runtime memory cost. The results show that the runtime memory cost becomes larger with the increase in γ. It is because with a larger γ, combos are more difficult to be found, and more vertex groups are generated and stored in Q, which contributes to a larger memory cost. However, it is noted that the *mc*s of combo searching are reasonable on all the tested datasets. Specifically, the total *mc*s are always smaller than 50 MB in the first twelve datasets, and the total *mc* of processing Dblp-4y is also smaller than 360 MB. Besides, the *mc* on a larger dataset may be lower than that on a smaller dataset. For example, the *mc* on Email-7d is lower than that on Email-1d when $\gamma = 0.6$, because the number of vertex groups traversed and stored in Q is the main factor that influences the *mc* as analyzed in Sect. 4.2,

Table 6 Memory cost of $\mathbb{T}$-cohesiveness evolution tracking

Memory cost on different datasets (MB)						
Group size	Col-1d	Col-7d	Col-30d	Email-1d	Email-7d	Email-30d
10	3.5	3.5	3.9	4.5	5.1	5.5
100	3.5	3.6	3.9	4.5	5.5	5.7
1000	3.6	3.6	4.2	4.7	5.5	5.8
Group size	Cont-1d	Cont-7d	Cont-30d	Math-1d	Math-7d	Math-30d
10	9.8	11.0	15.0	18.4	20.2	20.5
100	9.9	11.3	15.5	18.4	20.2	20.9
1000	10.1	11.4	16.1	18.8	20.4	21.4



(a) Col-1d (b) Email-1d (c) Cont-1d (d) Math-1d (e) Col-7d (f) Email-7d (g) Cont-7d

(h) Math-7d (i) Col-30d (j) Email-30d (k) Cont-30d (l) Math-30d (m) Dblp-4y

Fig. 10 The results of the memory cost experiments of combo searching

and more vertex groups may be pushed into Q when Email-1d is used in the experiments.

6.5 Ablation study

In this subsection, we illustrate the efficiency of our optimization method presented in Sect. 5. In detail, we denote Algorithm 3 by *CS*, and denote the algorithm without the optimization method by *CS-Opt* (i.e., removing Lines 18–19 in Algorithm 3). Experiments are conducted on Email-30d, and CF. 34 is used. Given $\gamma \in \{0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, 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 an hour, the algorithm is stopped and the time cost of this vertex is recorded as 1 h. The experimental results are shown in Fig. 11.

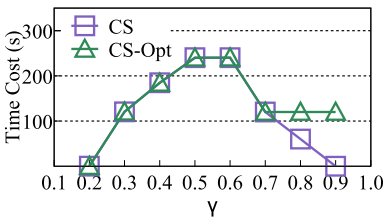


Fig. 11 Time cost of combo searching with *CS* and *CS-Opt*

As shown in Fig. 11, the experimental results suggest that our optimization method can accelerate combo searching significantly, and the optimization effect becomes better with the increase in γ. In detail, *CS* is more than 1000× 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 1 h and recorded as 1 h. 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 processed, and the time cost is reduced.

Moreover, in order to show the contributions of $\widehat{\zeta}_t$, $\widehat{\zeta}_{\text{inter}}$, and $\widehat{\zeta}_{\text{intra}}$, respectively, the incohesive vertex groups which can be pruned by $\widehat{\zeta}_t * \widehat{\zeta}_{\text{inter}} * \widehat{\zeta}_{\text{intra}}$, $\widehat{\zeta}_t * \widehat{\zeta}_{\text{inter}}$, $\widehat{\zeta}_t * \widehat{\zeta}_{\text{intra}}$, and $\widehat{\zeta}_{\text{inter}} * \widehat{\zeta}_{\text{intra}}$, respectively, are counted. Specifically, the dataset, configuration, and timeout threshold used are the same as those in the above ablation study. Since the optimization method is more efficient when $\gamma \geq 0.8$ as shown in Fig. 11, we report the experimental results when $\gamma \in \{0.8, 0.9\}$. Also, we report the results when $\gamma \in \{0.6, 0.65, 0.7, 0.75, 0.85\}$ for the sake of comprehensiveness. Besides, 30 query vertices are queried for each given γ, and 30 combos are required for each query vertex.

Table 7 The contribution of $\widehat{\zeta}_t$, $\widehat{\zeta}_{\text{inter}}$, and $\widehat{\zeta}_{\text{intra}}$

Dataset	γ	Number of pruned vertex groups			
		$\widehat{\zeta}_t * \widehat{\zeta}_{\text{inter}} * \widehat{\zeta}_{\text{intra}}$	$\widehat{\zeta}_t * \widehat{\zeta}_{\text{inter}}$	$\widehat{\zeta}_t * \widehat{\zeta}_{\text{intra}}$	$\widehat{\zeta}_{\text{inter}} * \widehat{\zeta}_{\text{intra}}$
Email-30d	0.60	8,392,893	6,427,673	7,176,577	5,415,003
	0.65	7,161,112	6,898,666	3,411,118	7,122,035
	0.70	7,497,129	7,495,904	4,025,776	7,482,066
	0.75	7,569,450	7,558,070	5,616,877	7,553,157
	0.80	7,826,573	7,825,710	6,068,796	7,825,710
	0.85	7,941,170	7,930,772	7,941,158	7,930,547
	0.90	8,454,598	8,452,278	8,454,596	8,454,447

As shown in Table 7, no matter which upper bound is removed, the efficiency of our pruning method decreases. It suggests that all the three upper bounds are necessary and effective. Besides, with a large γ (e.g., $\gamma = 0.9$), the difference between $\widehat{\zeta}_t * \widehat{\zeta}_{\text{inter}}$ (or $\widehat{\zeta}_t * \widehat{\zeta}_{\text{intra}}$, $\widehat{\zeta}_{\text{inter}} * \widehat{\zeta}_{\text{intra}}$) and $\widehat{\zeta}_t * \widehat{\zeta}_{\text{inter}} * \widehat{\zeta}_{\text{intra}}$ becomes small, which indicates that pruning with two upper bounds is already efficient when γ is large. It is because when γ is large, for many invalid vertex groups, the values of $\widehat{\zeta}_t * \widehat{\zeta}_{\text{inter}}$ (or $\widehat{\zeta}_t * \widehat{\zeta}_{\text{intra}}$, $\widehat{\zeta}_{\text{inter}} * \widehat{\zeta}_{\text{intra}}$) are already smaller than γ, and using two upper bounds is capable of pruning some vertex groups. Moreover, the number of incohesive vertex groups pruned by $\widehat{\zeta}_t * \widehat{\zeta}_{\text{inter}} * \widehat{\zeta}_{\text{intra}}$ when $\gamma = 0.6$ is larger than those when $0.65 \leq \gamma < 0.9$. The possible reason is that function *GetValidNeighbors* invoked at Line 7 of Algorithm 3 cannot find any valid neighbors of some query vertices when $\gamma \geq 0.65$, and Lines 9–20 are never executed when combos of these query vertices are queried. Therefore, no incohesive vertex groups are pruned in this process. However, when $\gamma = 0.6$, there are some valid neighbors for these query vertices, and some incohesive vertex groups are pruned. Then, the number of pruned incohesive vertex groups when $\gamma = 0.6$ is larger than those when $0.65 \leq \gamma < 0.9$. Nevertheless, the number of incohesive vertex groups pruned by $\widehat{\zeta}_t * \widehat{\zeta}_{\text{inter}} * \widehat{\zeta}_{\text{intra}}$ when $\gamma = 0.9$ is larger than that when $\gamma = 0.6$. The reason is that given $\gamma = 0.9$, much more incohesive vertex groups are pruned when the vertices having valid neighbors are queried, and $\widehat{\zeta}_t * \widehat{\zeta}_{\text{inter}} * \widehat{\zeta}_{\text{intra}}$ can prune more incohesive vertex groups in total.

Next, the experiments are conducted to confirm that the three terms of $\mathbb{T}$-cohesiveness (i.e., ζ_t, ζ_{inter}, and ζ_{intra}) are all effective. Please note that $\mathbb{T}$-cohesiveness is abbreviated as $\mathbb{T}_c$. Then, three new versions of $\mathbb{T}$-cohesiveness are proposed, namely $\mathbb{T}_c$-ζ_t, $\mathbb{T}_c$-ζ_{inter}, and $\mathbb{T}_c$-ζ_{intra}. They are obtained by removing ζ_t, ζ_{inter}, and ζ_{intra} from $\mathbb{T}_c$, respectively. CF. 15 is applied on Col-7d and Email-7d in the experiments, and we set $\gamma = 0.6$. For each dataset, given a γ, there are 30 query vertices, and 30 combos are required for each query vertex. The average values of T, R, and D of the obtained combos are reported in Fig. 2.

According to the first figures in Fig. 2a, b, the Ts of the combos obtained by $\mathbb{T}_c$-ζ_t are always larger than the Ts of the combos obtained by $\mathbb{T}_c$. It suggests that when ζ_t is removed, the obtained combos become less cohesive in temporal cohesiveness.

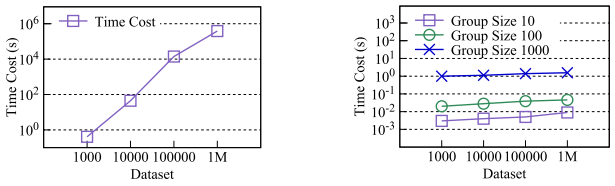
According to the second figures in Fig. 2a, b, the Rs of the combos obtained by $\mathbb{T}_c$-ζ_{inter} are always smaller than the Rs of the combos obtained by $\mathbb{T}_c$, and it suggests that when ζ_{inter} is removed, the obtained combos become less cohesive in inter-topological cohesiveness.

According to the third figures in Fig. 2a, b, the Ds of the combos obtained by $\mathbb{T}_c$-ζ_{intra} are always much smaller than the Ds of the combos obtained by $\mathbb{T}_c$. It suggests that when ζ_{intra} is removed, the obtained combos become less cohesive in intra-topological cohesiveness.

These results demonstrate that ζ_t, ζ_{inter}, and ζ_{intra} are all effective and have restrictions on the temporal, inter-topological, and intra-topological cohesiveness of combos, respectively. Please note that the combos obtained with $\mathbb{T}_c$-ζ_{inter} have small Ts. The reason may be that when the inter-topological cohesiveness is neglected, some subgraphs with few vertices can have a large value of $\mathbb{T}$-cohesiveness because of their large values of ζ_t and ζ_{intra}. Then, such subgraphs are obtained as combos, and their time spans are short because they only have a few temporal edges. Moreover, the combos obtained with $\mathbb{T}_c$-ζ_{inter} also have small Ds. The reason may be that some subgraphs have $\mathbb{T}$-cohesiveness values smaller than γ mainly due to their small values of ζ_{inter}. Then, when ζ_{inter} is removed, these subgraphs can have $\mathbb{T}$-cohesiveness larger than γ, although their ζ_{intra} values (and D values) are a bit small. Therefore, these subgraphs are obtained as combos, and the D value of $\mathbb{T}_c$-ζ_{inter} is smaller than that of $\mathbb{T}_c$.

6.6 Scalability evaluation

In this subsection, combo searching and $\mathbb{T}$-cohesiveness evolution tracking are performed on the four synthetic datasets (i.e., Syn_{1000}, Syn_{10000}, Syn_{100000}, and Syn_{1M}) to test their



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

Fig. 12 Results of scalability experiments. X-axis labels represent Syn_{1000}, Syn_{10000}, Syn_{100000}, and Syn_{1M}, resp

scalability. Specifically, CF. 1 is applied on these datasets, and γ is set to 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 cost of finding combos in Syn_{1M}. The average time cost on each dataset is reported in Fig. 12a. It is shown that the time cost grows almost linearly with the increase in 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 Fig. 12b. It is shown that the time cost of $\mathbb{T}$-cohesiveness evolution tracking increases slightly with the increase in the size of the temporal graph. It confirms the good scalability of our method.

6.7 Parameter sensitivity

The parameter sensitivity evaluation is conducted on Email-7d. As shown in Fig. 13, CF. 15 is the default configuration, $\gamma = 0.5$, 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 generated configuration, there are 30 query vertices, and 30 combos are required for each query vertex.

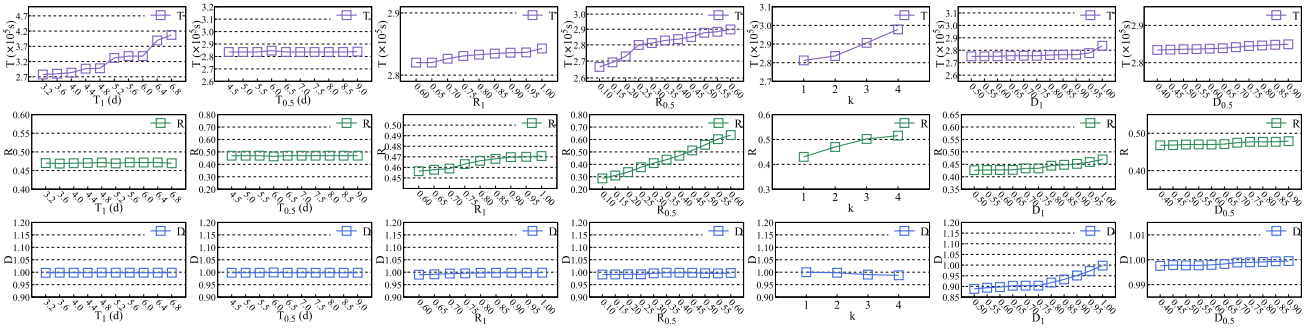
As shown in Fig. 13a, with the increase in T_1, the Ts of the obtained combos increase. The reason is that when T_1 becomes larger, the TC score of a subgraph with a fixed time span usually increases, and then the $\mathbb{T}$-cohesiveness value of the subgraph increases. It indicates that more subgraphs may be returned as combos, whose Ts are larger.

As shown in Fig. 13b, 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.

As shown in Fig. 13c, d, when the value of R_1 or $R_{0.5}$ increases, the original combos should contain more or fewer vertices to get larger R values to keep high InterTC scores, so that they are still combos. Since removing vertices from an original combo can only make it another original combo (which causes duplication) or less cohesive (otherwise, this subgraph should be obtained as an original combo because it is more cohesive than an original combo and is visited before an original combo), an original combo should contain more vertices and temporal edges, and the values of T increase as well.

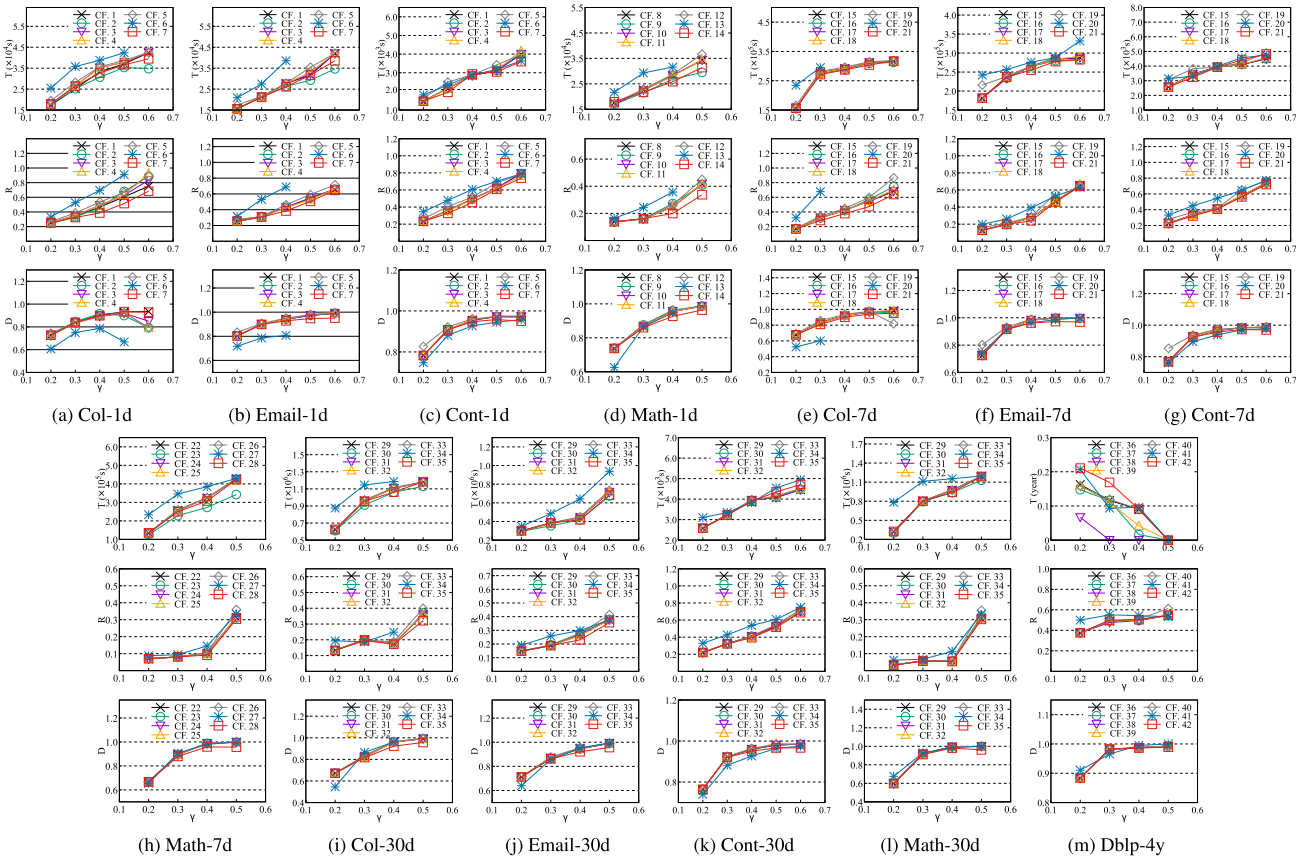
As shown in Fig. 13e, with the increase in k, according to Eq. 4, the density (i.e., D) of a subgraph usually decreases, and the IntraTC score of the subgraph also decreases. In order to make the Ds decrease slightly and get a large IntraTC score, the obtained combos contain more vertices to make their vertices have larger degrees. Then, the Ts and Rs of the combos also increase since more vertices and temporal edges are included.

As shown in Fig. 13f, g, 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.



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

Fig. 13 The results of parameter experiments on Email-7d



(a) Col-1d (b) Email-1d (c) Cont-1d (d) Math-1d (e) Col-7d (f) Email-7d (g) Cont-7d

(h) Math-7d (i) Col-30d (j) Email-30d (k) Cont-30d (l) Math-30d (m) Dblp-4y

Fig. 14 The results of the parameter experiments. In each subfigure, CF. x (black) is the base configuration, CF. $x+1$ (green) is generated by increasing T_1 (in Fig. 14m) or decreasing T_1 (in other figures); CF. $x+2$ (purple) is generated by decreasing $T_{0.5}$; CF. $x+3$ (orange) and CF. $x+4$ (gray) are generated by increasing R_1 and $R_{0.5}$, respectively; CF. $x+5$ (blue) is generated by increasing k; CF. $x+6$ (red) is generated by decreasing D_1. For Col-1d, Email-1d, Math-1d, Col-7d, and Col-30d, if $k = 3$ and $D_{0.5} = \frac{2}{3}$, the results when $\gamma = 0.4$, $\gamma = 0.5$ or $\gamma = 0.6$ are sometimes not reported, because 30 combos cannot be found for most query vertices in 30 min in those conditions (colour figure online)

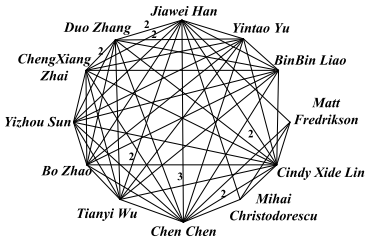
More sensitivity experiments are also conducted on the other twelve real-world datasets with CF. 1–CF. 42 in Table 4. The results are shown in Fig. 14, and the analysis and conclusions are similar to those presented above. The results on Dblp-4y (shown in Fig. 14m) are special cases. When a subgraph is required to be more cohesive to become a combo (e.g., when γ increases, $T_{0.5}$ decreases (CF. 38), and R_1 increases (CF. 39)), the combos obtained on Dblp-4y usually have smaller time spans to get a large value of $\mathbb{T}$-cohesiveness.

6.8 Case study

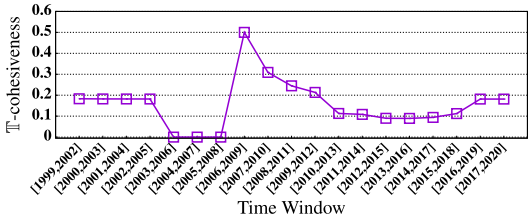
In this subsection, the two proposed time-topology analysis methods are applied on ***DBLP*** to demonstrate their good performances and justify their usefulness. In detail, combo searching is firstly performed, and $\mathbb{T}$-cohesiveness evolution tracking is then conducted.

Combo searching. The temporal edges with timestamps between year 2006 and 2009 are extracted from ***DBLP*** to generate a new temporal graph $\mathcal{G}_0$, and combo searching is then performed on $\mathcal{G}_0$ with CF. 36 and $\gamma = 0.5$. The query vertex is *Jiawei Han*. Figure 15a is one of the obtained combos, and its value of $\mathbb{T}$-cohesiveness is 0.50. The timestamps of temporal edges in Fig. 15a are all 2009 (and omitted), which indicates that these researchers only coauthored in 2009 during the period [2006, 2009].

The 12 researchers in Fig. 15a seem to have cohesive relationships in [2006, 2009] in reality. Specifically, *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



(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 Fig. 15a. 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.

Fig. 15 Case study on DBLP

Feb., 2010, these researchers may participate in this research project and coauthor papers in 2009. Then, they had cohesive relationships. Besides, *Binbin Liao* was a student in UIUC in 2008–2014 whose research area is computer vision. 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 mainly 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. It means that combo searching can find reasonable combos with practical meanings.

To perform quantitative analysis and show the usefulness of combo searching, 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 [20]. Eight quantitative metrics are used, i.e., d_r, d_t, O_v, deg_{avg}, $|V_s|$, D, $\mathbb{T}_c$, and clustering coefficient (abbr. CC) [14,30,32]. 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, each of which is outside $\mathcal{G}_s$, has a timestamp in the time period of $\mathcal{G}_s$, and 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 8. 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 $\mathbb{T}_c$.

$\mathbb{T}$-cohesiveness evolution tracking. $\mathbb{T}$-cohesiveness evolution tracking is then performed on ***DBLP***. The queried vertex group consists of all the 12 researchers in Fig. 15a, and every time window with a time span of 3 years is traversed. The results are shown in Fig. 15b. Because there are no edges among these researchers before 2002, the first time window used is [1999, 2002]. Fig. 15b 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 did not start 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 possible reasons: (1) *Binbin Liao* and the researchers in IBM do not coauthor with the researchers in DAIS after the project is finished; (2) Some researchers of DAIS graduated, and no longer coauthor with the other researchers; (3) The researchers left in DAIS coauthor papers in many years, and they are not cohesive in the time dimension.

Table 8 Quantitative comparison results

Alg.	Quantitative metrics							
	d_r	d_t	O_v	deg_{avg}	$\lvert V_s\rvert$	D	$\mathbb{T}_c$	CC
CS	**2**	**0y**	**37**	8.67	**12**	**1.0**	**0.50**	**0.90**
Dynamo	19	3y	3051	4.91	2990	0.93	0.27	0.42
KCA_3	3	3y	90,150	7.99	164,971	1.0	0.44	0.58
KCA_4	24	3y	151,703	**9.31**	116,126	1.0	0.37	0.62
EquiTruss	2	2y	121	7.69	13	1.0	0.24	0.85

7 Related work

The idea of time-topology analysis is partially inspired by the Time-Frequency Analysis [6] in signal processing, which analyzes a signal in both the time and frequency dimensions. A temporal graph also has two dimensions, i.e., the time dimension and the topology dimension, and then, we attempt to analyze temporal graphs in these two dimensions in this paper. To the best of our knowledge, this paper is the first work to analyze a temporal graph in both the time and topology dimensions. Therefore, in this section, we mainly introduce the related work of the time-topology analysis methods proposed based on $\mathbb{T}$-cohesiveness.

$\mathbb{T}$-cohesiveness evolution tracking traces the evolution of the cohesiveness of a group of vertices on a temporal graph. There are some previous studies that track the evolution of cohesive subgraphs. For example, Leskovec et al. [25] analyze the features of graph evolution. Takaffoli et al. [45] propose a method to detect community evolution in social networks. Zhang et al. [52] detect community structure in each snapshot of a temporal graph and track the evolution of subgraphs cohesive in the topology dimension. Please note that the temporal information is not utilized in these existing methods when the subgraphs are evaluated in each snapshot.

For combo searching, there are some previous studies finding cohesive subgraphs in normal graphs, attributed graphs, and temporal graphs. These studies are separately introduced in the rest of this section.

Cohesive subgraph finding in normal graphs. Many community detection algorithms are proposed to find cohesive subgraphs with the topological features of graphs (e.g., the degrees of the vertices). Specifically, Raghavan et al. [38] proposed the algorithm of label propagation (LPA) in 2007. Leung et al. [26] proposed HANP with "hop attenuation" and "node preference" to further improve the performance of LPA. Blondel et al. [3] proposed a well-known hierarchical community detection algorithm named Louvain in 2008. It detects communities in a bottom-up style by combining communities to optimize the modularity. Some more commonly used community detection algorithms are summarized, classified, and evaluated in [16] and [46].

Note that, community detection algorithms focus on the topology of the whole graph and attempt to divide the original graph into subgraphs that are cohesive, respectively. However, sometimes cohesive subgraphs containing specific vertices are wanted, and the local structure of the vertices is the focus in that condition. Therefore, the community search problem is proposed, and numerous algorithms are presented [7,13–15,21,42,48]. Specifically, a subgraph is returned as a cohesive subgraph in the community search problem, if it contains the query vertex and satisfies a specific structural constraint such as k-clique [10], k-core [8,15,24,41], and k-truss [5,20].

Cohesive subgraph finding in attributed graphs. Besides normal graphs, there are also some community detection and community search algorithms that find cohesive subgraphs in attributed graphs [4,9,11,22,47,49,51]. For instance, Cai et al. [4] propose that each vertex in a community should contain all the query attributes. ATAC [11] associates every vertex with a set of activities and requires the vertices in a community to participate in many activities related to the queried topics. Since these algorithms deal with attributed graphs whose vertices have attributes while the edges do not, when these methods are applied on temporal graphs, where the temporal edges have timestamps, they cannot exploit the temporal information to find cohesive subgraphs and usually have poor performances.

Cohesive subgraph finding in temporal graphs. Some existing studies can find cohesive subgraphs in temporal graphs to some extent [2,12,18,34,43,52]. A typically related problem is the dynamic community detection problem that discovers communities in temporal graphs. A considerable amount of algorithms have been proposed for it. These studies can be divided into two-stage methods [12,43] and evolutionary clustering methods [18,34,52]. The two-stage methods always follow a two-stage procedure [12]. First, topologically cohesive subgraphs (communities) are obtained independently for each temporal snapshot. Second, the obtained subgraphs are compared and matched to obtain their evolutions through temporal adjacent analysis. The evolutionary clustering methods detect communities using an incremental manner where communities detected in a snapshot are affected by the ones detected in the last snapshot to ensure temporal smoothness [52].

However, these existing dynamic community detection algorithms only search for densely connected subgraphs on network snapshots [39], which may lose the topological information in the previous and next snapshots. Besides, the temporal information is only exploited to divide a temporal graph into snapshots and is not properly utilized.

8 Conclusions

In this paper, time-topology analysis on temporal graphs is proposed to discover interesting patterns. In detail, Temporal and Topological Cohesiveness (abbr. $\mathbb{T}$-cohesiveness) is first proposed to evaluate the cohesiveness of temporal graphs in both the topology and time dimensions. Specifically, for a graph with a large value of $\mathbb{T}$-cohesiveness, its vertices should be densely connected, and there should be much more intra-edges than inter-edges. Besides, the timestamps of the edges in the graph should be within a short period of time. Then, $\mathbb{T}$-cohesiveness is applied to perform time-topology analysis on temporal graphs. Two types of analyses, i.e., $\mathbb{T}$-cohesiveness evolution tracking and combo searching, are

proposed, and their corresponding algorithms are presented. Moreover, in order to perform combo searching more efficiently, an optimization method is proposed to prune the vertex groups whose supergraphs can never be a combo. Finally, the experimental results on both real-world and synthetic datasets demonstrate that our proposed time-topology analysis methods are superior to Dynamo and k-core algorithms. The ablation studies suggest that the optimization method is efficient and can reduce the time cost of combo searching significantly. In the future, we will consider utilizing $\mathbb{T}$-cohesiveness to perform more kinds of time-topology analysis on temporal graphs, so that $\mathbb{T}$-cohesiveness can be applied to solve more practical problems. Besides, the method to generalize $\mathbb{T}$-cohesiveness to deal with directed graphs will be studied.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (No. 61872207) and Baidu Inc.

References

1. Akbas, E., Zhao, P.: Truss-based community search: a truss-equivalence based indexing approach. PVLDB **10**(11), 1298–1309 (2017)
2. Belth, C., Zheng, X., Koutra, D.: Mining persistent activity in continually evolving networks. In: SIGKDD, pp. 934–944 (2020)
3. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. Theory Exp. **10**, P10008 (2008)
4. Cai, T., Li, J., Haldar, N.A.H., Mian, A., Yearwood, J., Sellis, T.: Anchored vertex exploration for community engagement in social networks. In: ICDE, pp. 409–420 (2020)
5. Cohen, J.: Trusses: cohesive subgraphs for social network analysis. Natl. Secur. Agency Tech. Rep. **16**, 3–29 (2008)
6. Cohen, L.: Time-Frequency Analysis, vol. 778. Prentice Hall, Hoboken (1995)
7. Cui, W., Xiao, Y., Wang, H., Lu, Y., Wang, W.: Online search of overlapping communities. In: SIGMOD/PODS'13, pp. 277–288 (2013)
8. Cui, W., Xiao, Y., Wang, H., Wang, W.: Local search of communities in large graphs. In: SIGMOD/PODS'14, pp. 991–1002 (2014)
9. Dang, T., Viennet, E.: Community detection based on structural and attribute similarities. In: ICDS, pp. 7–12 (2012)
10. Danisch, M., Balalau, O., Sozio, M.: Listing k-cliques in sparse real-world graphs. In: WWW '18, pp. 589–598 (2018)
11. Das, B.C., Anwar, M.M., Bhuiyan, M.A.A., Sarker, I.H., Alyami, S.A., Moni, M.A.: Attribute driven temporal active online community search. IEEE Access **9**, 93976–93989 (2021)
12. Dhouioui, Z,. Akaichi, J.: Tracking dynamic community evolution in social networks. In: ASONAM 2014, pp. 764–770. IEEE (2014)
13. Dong, Z., Huang, X., Yuan, G., Zhu, H., Xiong, H.: Butterfly-core community search over labeled graphs. Proc. VLDB Endow. **14**(11), 2006–2018 (2021)
14. Fang, Y., Huang, X., Qin, L., Zhang, Y., Zhang, W., Cheng, R., Lin, X.: A survey of community search over big graphs. VLDB J. **29**(1), 353–392 (2020)
15. Fang, Y., Yang, Y., Zhang, W., Lin, X., Cao, X.: Effective and efficient community search over large heterogeneous information networks. PVLDB **13**(6), 854–867 (2020)
16. Fortunato, S., Hric, D.: Community detection in networks: a user guide. Phys. Rep. **659**, 1–44 (2016)
17. Giatsidis, C., Thilikos, D.M., Vazirgiannis, M.: D-cores: measuring collaboration of directed graphs based on degeneracy. Knowl. Inf. Syst. **35**(2), 311–343 (2013)
18. Guo, C., Wang, J., Zhang, Z.: Evolutionary community structure discovery in dynamic weighted networks. Physica A **413**, 565–576 (2014)
19. Han, W., Miao, Y., Li, K., Wu, M., Yang, F., Zhou, L., Prabhakaran, V., Chen, W., Chen. E.: Chronos: a graph engine for temporal graph analysis. In: Proceedings of the Ninth European Conference on Computer Systems, pp. 1–14 (2014)
20. Huang, X., Cheng, H., Qin, L., Tian, W., Yu, J.X.: Querying k-truss community in large and dynamic graphs. In: SIGMOD/PODS'14, pp. 1311–1322 (2014)
21. Huang, X., Lakshmanan, L.V., Xu, J.: Community Search Over Big Graphs, vol. 14. Morgan & Claypool Publishers, San Rafael (2019)
22. Jia, C., Li, Y., Carson, M.B., Wang, X., Yu, J.: Node attribute-enhanced community detection in complex networks. Sci. Rep. **7**(1), 1–15 (2017)
23. Kemal, M.U.: Anti-money laundering regulations and its effectiveness. J. Money Laund. Control **17**(4), 416–427 (2014)
24. Khaouid, W., Barsky, M., Srinivasan, V., Thomo, A.: K-core decomposition of large networks on a single pc. PVLDB **9**(1), 13–23 (2015)
25. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: densification and shrinking diameters. TKDD **1**(1), 2-es (2007)
26. Leung, I.X.Y., Pan, H., Lio, P., Crowcroft, J.: Towards real-time community detection in large networks. Phys. Rev. E Stat. Nonlinear Soft Matter Phys. **79**(6 Pt 2), 066107 (2009)
27. Levi, M., Reuter, P.: Money laundering. Crime Justice **34**(1), 289–375 (2006)
28. Li, R.H., Su, J., Qin, L., Yu, J.X., Dai, Q.: Persistent community search in temporal networks. In: ICDE, pp. 797–808. IEEE (2018)
29. Lima, R.S., Serrano, A.L.M., Imoniana, J.O., Cupertino, C.M.: Identifying financial patterns of money laundering with social network analysis: a Brazilian case study. J. Money Laund. Control **25**(1), 118–134 (2021)
30. Liu, B., Zhang, F., Zhang, W., Lin, X., Zhang, Y. Efficient community search with size constraint. In: ICDE, pp. 97–108. IEEE (2021)
31. Liu, Q., Zhu, Y., Zhao, M., Huang, X., Xu, J., Gao, Y.: Vac: vertex-centric attributed community search. In: ICDE, pp. 937–948. IEEE (2020)
32. Luce, R.D., Perry, A.D.: A method of matrix analysis of group structure. Psychometrika **14**(2), 95–116 (1949)
33. Luo, J., Cao, X., Xie, X., Qu, Q., Xu, Z., Jensen, C.S.: Efficient attribute-constrained co-located community search. In: ICDE, pp. 1201–1212. IEEE (2020)
34. Márquez, R., Weber, R., De Carvalho, A.C.: A non-negative matrix factorization approach to update communities in temporal networks using node features. In: ASONAM, pp. 728–732. IEEE (2019)
35. Panzarasa, P., Opsahl, T., Carley, K.M.: Patterns and dynamics of users' behavior and interaction: network analysis of an online community. J. Am. Soc. Inform. Sci. Technol. **60**(5), 911–932 (2009)
36. Paranjape, A., Benson, A.R., Leskovec, J.: Motifs in temporal networks. In: WSDM, pp. 601–610 (2017)
37. Pyle, D.H.: Bank Risk Management: Theory. Risk Management and Regulation in Banking, pp. 7–14. Springer, Berlin (1999)
38. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E **76**(3), 036106 (2007)
39. Rossetti, G., Cazabet, R.: Community discovery in dynamic networks: a survey. CSUR **51**(2), 1–37 (2018)

40. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI, pp. 4292–4293 (2015)
41. Seidman, S.B.: Network structure and minimum degree. Soc. Netw. **5**(3), 269–287 (1983)
42. Sozio, M., Gionis, A.: The community-search problem and how to plan a successful cocktail party. In: KDD '10, pp. 939–948 (2010)
43. Sun, J., Faloutsos, C., Papadimitriou, S., Yu, P.S.: Graphscope: parameter-free mining of large time-evolving graphs. In: KDD '07, pp. 687–696 (2007)
44. Sun, X., Feng, W., Liu, S., Xie, Y., Bhatia, S., Hooi, B., Wang, W., Cheng, X.: Monlad: money laundering agents detection in transaction streams. In: WSDM, pp. 976–986 (2022)
45. Takaffoli, M., Sangi, F., Fagnan, J., Zäıane, O.R.: Community evolution mining in dynamic social networks. Procedia Soc. Behav. Sci. **22**, 49–58 (2011)
46. Wang, M., Wang, C., Yu, J.X., Zhang, J.: Community detection in social networks: an in-depth benchmarking study with a procedure-oriented framework. PVLDB **8**(10), 998–1009 (2015)
47. Wang, X., Jin, D., Cao, X., Yang, L., Zhang, W.: Semantic community identification in large attribute networks. In: AAAI, pp. 265–271 (2016)
48. Wu, Y., Jin, R., Li, J., Zhang, X.: Robust local community detection: on free rider effect and its elimination. PVLDB **8**(7), 798–809 (2015)
49. Wu, Z., Lu, Z., Ho, S.Y.: Community detection with topological structure and attributes in information networks. TIST **8**(2), 1–17 (2016)
50. Zhang, F., Zhang, Y., Qin, L., Zhang, W., Lin, X.: When engagement meets similarity: efficient (k, r)-core computation on social networks. PVLDB **10**(10), 998–1009 (2017)
51. Zhe, C., Sun, A., Xiao, X.: Community detection on large complex attribute network. In: KDD '19, pp. 2041–2049 (2019)
52. Zhuang, D., Chang, J.M., Li, M.: Dynamo: dynamic community detection by incrementally maximizing modularity. TKDE **33**(5), 1934–1945 (2019)