Social Network Analysis and Mining (2024) 14:54
https://doi.org/10.1007/s13278-024-01207-y

ORIGINAL ARTICLE

A two-phase approach for enumeration of maximal (Δ, γ)-cliques of a temporal network

Suman Banerjee[1] · **Bithika Pal**[2]

Received: 27 September 2023 / Revised: 6 January 2024 / Accepted: 19 January 2024

Abstract
A *Temporal Network* is often used to model a time-varying relationship among a group of agents. It is typically represented as a collection of triplets of the form (u, v, t) that denote the interaction between the agents u and v at time t. For analyzing structural patterns of such a network, the notion of (Δ, γ)-*cliques* has been introduced in one of our previous studies. A (Δ, γ)-clique of a temporal network is a vertex subset–time interval pair such that there exist at least γ links between every pair of vertices of the vertex set in each Δ duration of the time interval. In this paper, we propose a two-phase approach for enumerating maximal (Δ, γ)-cliques present in a temporal network. The proposed methodology is broadly divided into two phases. In the first phase, each temporal link is processed for constructing (Δ, γ)-clique(s) with maximum duration. In the second phase, these initial cliques are expanded by vertex addition to form the maximal cliques. By sequential arguments, we show that the proposed methodology correctly enumerates all the maximal (Δ, γ)-cliques. A comprehensive analysis of the running time and space requirement of the proposed methodology has been carried out. From the experimentation performed on 5 datasets, we observe that the proposed methodology enumerates all the maximal (Δ, γ)-cliques efficiently, particularly when the dataset is sparse. As a special case ($\gamma = 1$), the proposed methodology is also able to enumerate $(\Delta, 1) \equiv \Delta$-cliques in much less time compared to the existing methods.

Keywords Temporal network · Enumeration algorithm · (Δ, γ)-clique

1 Introduction

A network (also called graph) is a mathematical object which is used extensively to represent a *binary relation* among a group of agents. Analyzing such networks for different structural patterns remains an active area of study in different domains including *Computational Biology* (Hulovatyy et al. 2015), *Social Network Analysis*, *Computational Epidemiology* (Masuda and Holme 2017), Criminal Network Analysis (Ficara et al. 2021), and many more. Among many, one such structural pattern is the maximally connected subgraphs, which are popularly called *cliques*. Finding the maximum cardinality clique in a given network is a well-known *NP-Complete* Problem (Garey and Johnson 2002). However, in network analysis, perspective more general problem is not finding the maximum size clique, but also to enumerate all the maximal cliques present in the network. Bron and Kerbosch (1973) first proposed an enumeration algorithm for maximal cliques in the network which forms the foundation of study on this problem. Later, there were advancements for this problem for different types of networks (Cheng et al. 2012; Eppstein et al. 2013).

Real-world networks from biological to social are *time-varying*, which means that the existence of an edge between any two agents changes with time. Temporal networks (Holme and Saramäki 2012) (also known as *link streams* or *time-varying networks*) are the mathematical objects used to formally represent the time-varying relationships. For these types of networks, a natural

Both the authors have contributed equally in this study. A small part of this work has been previously published as Banerjee and Pal (2021).

✉ Suman Banerjee
suman.banerjee@iitjammu.ac.in

Bithika Pal
bithikapal@iitkgp.ac.in

[1] Department of Computer Science and Engineering, Indian Institute of Technology Jammu, Jagti, India

[2] Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India

Published online: 08 March 2024

supplement of clique is the *temporal clique* which consists of two things: a subset of the vertices and a time interval. In this direction, recently, Viard et al. (2016) put forward the notion of Δ-clique, where a vertex subset along with a time interval is said to be a Δ-clique if every vertex pair from that set has at least a single edge in every Δ duration within the time interval. Next, we report the existing studies on clique enumeration on networks.

As mentioned previously, a temporal network consists of a set of agents and a time-varying relationship. Now, the following questions are essential to understand the contact pattern among them: which subset of agents comes in contact very frequently among each other? Given a time duration, how many times do they contact each other? etc. The frequency of communication also adds another dimension of information to their relationship strength. Motivated by such questions, recently, the notion of Δ-clique has been extended to (Δ, γ)-*cliques*, which is basically a vertex subset and time interval pair in which each pair of vertices of the subset has at least γ interactions in every Δ duration within the time interval. We propose a different approach for listing out all the maximal (Δ, γ)-cliques contained in a temporal network. The main contributions of this paper are as follows:

- In this paper, we propose a different approach for listing out maximal (Δ, γ)-cliques that are there in a temporal network.
- By drawing sequential arguments, we prove the correctness of the proposed methodology.
- A detailed analysis of the proposed methodology has been done to understand its computational time and space requirement.
- The proposed methodology has been implemented with five publicly available temporal network datasets to bring out nontrivial insights about contact patterns and compare the efficiency of the proposed methodology with the existing one.
- Also, a set of experiments has been conducted to show that the proposed methodology of maximal (Δ, γ)-clique enumeration can also be efficiently used for enumerating maximal Δ-clique as well (By putting $\gamma = 1$).

The remaining portion of this article is arranged in the following way: Sect. 2 describes some relevant studies from the literature. Section 3 discusses some preliminary concepts regarding temporal networks and formally defines the maximal (Δ, γ)-clique enumeration problem. Section 4 contains the proposed enumeration technique with its detailed analysis, proof of correctness, and an illustrative example. Section 5 describes an experimental evaluation of the proposed methodology. Finally, Sect. 6 concludes this study and gives future directions.

2 Related work

In recent times, mining and analysis of temporal networks have become an active area of research as most of the real-world networks from social to biological are temporal in nature (Rozenshtein and Gionis 2019). Several problems including community analysis (Qin et al. 2020), finding matching (Zschoche 2022), finding separators (Zschoche et al. 2020), coloring (Mertzios et al. 2021), traversal (Byun et al. 2019), etc., have been studied in the context of temporal graphs. As per the title of the paper, here, we discuss the literature related to clique enumeration of static and followed by temporal graphs.

The problem of maximal clique enumeration is a classic computational problem on network algorithms and has been extensively studied on static networks. Akkoyunlu (1973) was the first to propose an algorithm for this problem. Later, Bron and Kerbosch (1973) introduced a recursive approach for the maximal clique enumeration problem. These two studies are the foundations on maximal clique enumeration and trigger a huge amount of research due to many practical applications from computational biology to spatial data analytics (Al-Naymat 2008) and Bhowmick and Seah (2015). In the past two decades, several methodologies have been developed for enumerating maximal cliques in different computational paradigms, and different kinds of networks, such as in sparse graphs (Eppstein et al. 2013; Manoussakis 2019), in large networks (Cheng et al. 2010, 2011; Rossi et al. 2014), in map-reduce framework (Hou et al. 2016; Xiang et al. 2013), in uncertain graphs (Mukherjee et al. 2016; Zou et al. 2010; Dai et al. 2022), in parallel computing framework (Chen et al. (2016); Rossi et al. (2015); Schmidt et al. (2009)), in signed networks (Chen et al. 2020), in temporal networks (Banerjee and Pal 2022), and many more (Dai et al. 2023; Manoussakis 2023).

Though there are many existing studies on maximal clique enumeration on static networks, the literature on temporal graphs is limited. Viard et al. (2015) proposed an enumeration algorithm for the maximal Δ-clique of a temporal network. They did a detailed analysis of contact relationships among a group of students, based on their introduced methodology. They were able to show that their analysis draws deeper insights of their communication pattern (Viard et al. 2015). Later, Himmel et al. (2016) proposed a different approach for the maximal Δ-clique enumeration problem. Their methodology is based on the *Bron–Kerbosch Algorithm* for maximal clique enumeration in static graphs. Their methodology is better in both of the following aspects: theoretically (measured in terms of worst case computational complexity analysis) as well as practically (measured in terms of computational time

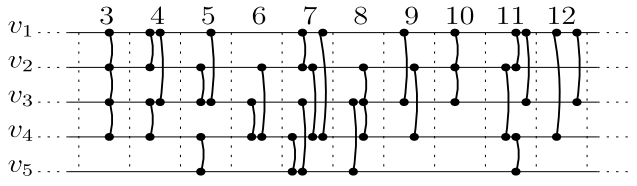

Fig. 1 Link stream representation of a temporal network

when the algorithm is implemented with real-world datasets). Molter et al. (2019) introduced the notion of *isolation* in clique enumeration of a time-varying graph. They developed fixed parameter enumeration algorithms based on different notions of isolation employing the parameter "degree of isolation." Viard et al. (2018) generalized the notion to contact with duration and introduced the concept of Δ-clique with duration. They also proposed an algorithm for enumerating such cliques present in a temporal network. Bentert et al. (2019) studied the maximal Δ-Plex enumeration problem. Recently, Banerjee and Pal (2019) proposed an enumeration algorithm for maximal (Δ, γ) *-cliques* present in a time-varying graph. The method initializes a clique for each link in the temporal network and expands its duration and cardinality to find the maximal cliques. In this work, we propose a two-phase approach by generating the initial cliques as duration-wise maximal cliques, which significantly reduces the number of intermediate cliques generated in the enumeration process. As far as we know, other than the last one, there is no other work available which studies (Δ, γ)-*cliques*.

3 Background and problem definition

In this section, we present some preliminary concepts to understand the problem, that we work on this paper and the proposed solution methodology. In a *temporal network*, its edges are marked with the corresponding occurrence timestamp(s). Formally, it is stated in Definition 1.

Definition 1 (Temporal Network) A temporal network is defined as $\mathcal{G}(V, E, \mathcal{T})$, where $V(\mathcal{G})$ is the set of vertices of the network, and $E(\mathcal{G})$ is the set of edges among them. $\mathcal{T}$ is the mapping that maps each edge of the graph to its occurrence time stamp(s), i.e., $\mathcal{T}: E(\mathcal{G}) \longrightarrow 2^T \backslash \emptyset$ where $T = \{1, 2, \ldots, \mathbb{T}\}$ is the set of discrete time stamps in which the network is observed.

A temporal network can be represented in two ways. One approach is to represent a temporal network using the link stream model where we show the relationships among the entities over the time horizon. The other approach is the time stamp-wise snapshot graph representation. In this approach, a temporal network is represented as a collection of static graphs overtime stamps. Figures 1 and 2 show the representation of the same temporal network in the form of link stream model and snapshot graph representation model, respectively. In the rest of the paper, we consider that the temporal network is represented in the link stream model.

As just mentioned, Fig. 1 shows a temporal graph with five vertices and 29 edges, where edges are shown in the time horizon. In temporal network analysis, it is assumed that the network changes its topology in discrete time steps. So, starting at time t, if the network is observed in every dt time difference till $t^{'}$, the time instances are $\mathbb{T} = \{t, t + dt, t + 2dt, \ldots, t^{'}\}$. In the rest of our study, we assume, $t, t^{'} \in \mathbb{Z}^+$ and $dt = 1$. The difference between the beginning and ending time stamp, i.e., $t^{'} - t$ is called as the *Lifetime of the Network*. In the temporal network $\mathcal{G}$, if there is an edge between two vertices v_i and v_j at time $t^{''}$, then it is symbolized as $(v_i, v_j, t^{''})$, signifying that there is a contact between u and v at time $t^{''}$. For some $t^{''} \in \mathbb{T}$ if $(u, v, t^{''}) \in E(\mathcal{G})$, then we say that there exists a static edge between v_i and v_j. The *frequency* of an edge is defined as how many distinct time stamps $t^{''}$ are there in the time span $\mathbb{T}$ such that $(v_i, v_j, t^{''}) \in E(\mathcal{G})$ and denoted as $f_{(v_i v_j)}$, i.e., $f_{(v_i v_j)} = |\{t^{''} \in \mathbb{T} : (v_i, v_j, t) \in E(\mathcal{G})\}|$. If there does not exist any $t^{''} \in \mathbb{T}$ such that $(v_i, v_j, t^{''}) \notin E(\mathcal{G})$, then we say that $f_{(v_i, v_j)} = 0$. In the rest of our study, we work with undirected temporal network, i.e., there is no difference between $(v_i, v_j, t^{''})$ and $(v_j, v_i, t^{''})$.

In a static network, a subset of vertices, where every pair is adjacent, is known as a *clique*. The size of the clique is defined as the number of vertices it contains. A clique is said to be maximal if it is not part of another clique of larger size. In one of our recent studies, we introduced the notion of (Δ, γ)-clique by extending the concept of Δ-clique and incorporating an additional

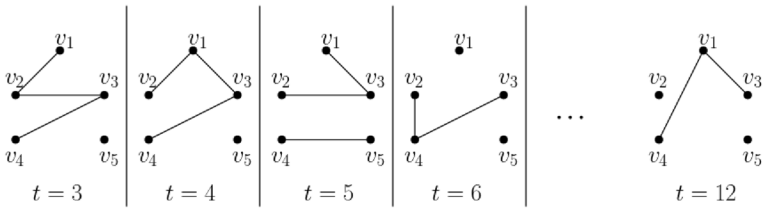

Fig. 2 Snapshot graph representation of the temporal network shown in Figure 1

parameter γ as a frequency threshold. This is stated in Definition 2.

Definition 2 ((Δ,γ)-clique) (Banerjee and Pal 2019) Given a temporal network $\mathcal{G}(V,E,\mathcal{T})$, time duration Δ, and a frequency threshold $\gamma \in \mathbb{Z}^+$, a (Δ,γ)-clique of $\mathcal{G}$ is a tuple consisting of vertex subset, and time interval, i.e., $(\mathcal{X},[t_a,t_b])$ where $\mathcal{X} \subseteq V(\mathcal{G}), |\mathcal{X}| \geq 2$, and $[t_a,t_b] \subseteq \mathbb{T}$. Here $\forall v_i, v_j \in \mathcal{X}$ and $\tau \in [t_a, max(t_b-\Delta, t_a)]$, there must exist at least γ number of edges, i.e., $(v_i,v_j,t_{ij}) \in E(\mathcal{G})$ and $f_{(v_iv_j)} \geq \gamma$ with $t_{ij} \in [\tau, min(\tau+\Delta, t_b)]$. Here, $f_{(v_iv_j)}$ denotes the frequency of the static edge (v_i,v_j).

In a static graph $G(V,E)$, a maximal clique is formed as $\mathcal{S} \subset V(G)$, if for each $v \in V(G)\backslash\mathcal{S}$, $\mathcal{S}\cup\{v\}$ is not a clique. Now, as the (Δ,γ)-clique is defined in the setting of temporal networks, its maximality depends on two parameters: One is the *cardinality* (referred to as inclusion-wise maximality) and the other one is the *time interval* (referred to as temporally maximal). We introduce the maximality conditions for an arbitrary (Δ,γ)-clique in Definition 3 considering both the factors.

Definition 3 (Maximal (Δ,γ)-clique) Given a temporal network $\mathcal{G}(V,E,\mathcal{T})$ and a (Δ,γ)-clique $(\mathcal{X},[t_a,t_b])$ of $\mathcal{G}$, $(\mathcal{X},[t_a,t_b])$ will be maximal if none of the following is true.

- $\exists v \in V(\mathcal{G}) \setminus \mathcal{X}$ such that $(\mathcal{X}\cup\{v\},[t_a,t_b])$ is a (Δ,γ) -clique.
- $(\mathcal{X},[t_a-1,t_b])$ is a (Δ,γ)-clique. This applies only if $t_a-1 \geq t$.
- $(\mathcal{X},[t_a,t_b+1])$ is a (Δ,γ)-clique. This applies only if $t_b+1 \leq t'$.

In this paper, we study the problem of listing out all the maximal (Δ,γ)-cliques of a given temporal network, which we call as the maximal (Δ,γ)-clique enumeration problem defined next.

Definition 4 (Maximal (Δ,γ)-clique enumeration problem) Given a temporal network $\mathcal{G}(V,E,\mathcal{T})$, Δ, and γ the maximal (Δ,γ)-clique enumeration problem asks to list out all the maximal (Δ,γ)-cliques (as mentioned in Definition 3) present in $\mathcal{G}$.

Table 1 lists out all the symbols and notations used in this paper along with their interpretation. Next, we proceed to describe the proposed enumeration methodology for maximal (Δ,γ)-cliques.

Table 1 Symbols and notations used in this paper

Symbol	Interpretation
$\mathcal{G}(V,E,\mathcal{T})$	A temporal network
$V(\mathcal{G}), E(\mathcal{G})$	Vertex set and link set of $\mathcal{G}$
$\mathbb{T}$	
n, m	Number of vertices and links of $\mathcal{G}$
(v_i, v_j, t'')	Any arbitrary link of $\mathcal{G}$
$f_{(v_i,v_j)}$	The frequency of the edge (v_i, v_j)
$(\mathcal{X},[t_a,t_b])$	An arbitrary (Δ,γ)-clique of $\mathcal{G}$
t^f, t^l	First and last occurrence of the edge under consideration
t^i	i-th occurrence of the edge under consideration
f^γ, l^γ	Last γ-th occurrence of the edge under consideration
$G(V,E')$	Static graph of the temporal network
$V(G), E(G)$	Vertex set and edge set of G
$\mathcal{N}_G(\mathcal{X})$	Neighborhood of the vertex set $\mathcal{X}$ in the graph G
$f_{\max}$	Maximum frequency among all edges of G
$\mathcal{D}, \mathcal{D}_{\text{Temp}}$	Dictionaries used in Algorithm 1 and
$\mathcal{T}_{(u,v)}$	Time stamps of the edge (u, v)
$\mathcal{C}_T^l, \mathcal{C}^{\mathcal{T}_1}, \mathcal{C}^{\mathcal{T}_2}$	Different lists used in Algorithm 1 and
$\mathcal{C}_T$	The maximal (Δ,γ)-clique set of $\mathcal{G}$
$len(\mathcal{C}_T)$	Length of the list $\mathcal{C}_T$
$\mathbb{Z}^+$	The set of positive integers

4 Proposed enumeration technique

As stated earlier, the proposed methodology is broadly divided into two steps, and each of them is described in the following two subsections. The broad idea of the proposed enumeration process is as follows: Given all the links with time duration of the temporal network, initially, we find out the maximal cliques of cardinality two. Next, taking these duration-wise maximal cliques, we add vertices into the clique without violating the definition of (Δ,γ)-cliques.

4.1 Stretching phase (initialization)

Algorithm 1 describes the initialization process of the proposed methodology. For a given temporal network $\mathcal{G}$, initially, we construct the dictionary $\mathcal{D}_e$ with the static edges as the *keys*, and correspondingly, the occurrence time stamps are the *values*. By the definition of (Δ,γ)-clique, if the end vertices of an edge are part of the same clique, then the edge has to occur at least γ times in the link stream. Hence, for each static edge (u, v) of $\mathcal{G}$, if its frequency is at least γ, it is processed further. The occurrence time stamps of (u, v) are fed into the list $\mathcal{T}_{(u,v)}$. A temporary list, *Temp*, is created to store each current processing timestamp from $\mathcal{T}_{(u,v)}$ with its previous occurrences, till it has maintained (Δ,γ)-clique property. Now, the for loop from Lines 8–32 computes all the (Δ,γ)-cliques with maximum duration where $\{u,v\}$ is the vertex set. During the processing of $\mathcal{T}_{(u,v)}$, any one of the following two cases can happen. In the first

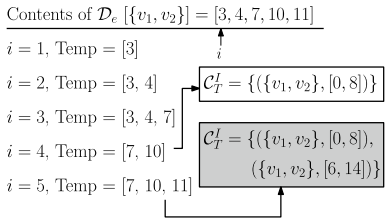

Fig. 3 An illustrative example of algorithm 1 using the temporal network of Fig. 1, for the link (v_1, v_2) with $\Delta = 4$ and $\gamma = 2$. All the temporally maximal (Δ, γ)-cliques of the vertex pair $\{v_1, v_2\}$ are kept in the initialized clique set, $\mathcal{C}_T^I$, marked in gray color

case, if the current length of *Temp* is less than γ, the difference between the current timestamp from $\mathcal{T}_{(u,v)}$ and the first entry of *Temp* is checked (Line 10). Now, if the difference is less than or equal to Δ, current timestamp is appended in *Temp*. Otherwise, all the previous timestamps that have occurred within past Δ duration from the current timestamp are added in *Temp* (Line 14). This process basically checks Δ timestamps backward from each occurrence times of the static edge (u, v). In the second case, when the current length of *Temp* is greater than or equal to γ, it is checked whether the current processing time from $\mathcal{T}_{(u,v)}$ falls within the interval of (last γ-th occurrence time + 1) to (last γ-th occurrence time + 1 + Δ). Now, if it is true, the current timestamp is appended in *Temp*. It can be easily observed that this appending is done if at least the consecutive γ occurrences are within each Δ duration. Otherwise, the clique is added in $\mathcal{C}_T^I$ with the vertex set $\{u, v\}$ and time interval $[t_a, t_b]$ (Line 22), where t_a is the Δ ahead timestamp from the first γ-th entry in *Temp*, and t_b is the Δ onwards timestamp from the last γ-th entry in *Temp*. Next, all the previous timestamps that have occurred within past Δ duration from the current timestamp are added in *Temp* as before (Line 24). It allows to consider overlapping cliques. Now, this may happen when we process the last occurrence from $\mathcal{T}_{(u,v)}$, it is added in *Temp*. However, no clique can be added by the condition of 9–26 if the length of *Temp* is greater than or equal to γ. This situation is handled by Lines 27–31. This process is iterated for each key from the dictionary $\mathcal{D}_e$. Now, we present lemmas that together they will help to argue the correctness of the proposed methodology. An illustrative example of Algorithm 1 for one link is shown in Fig. 3.

Algorithm 1 Stretching phase of the (Δ, γ)-clique enumeration

Data: The temporal network $\mathcal{G}(V, E, \mathcal{T})$, Δ, $\gamma \in \mathbb{Z}^+$.
Result: The initial clique set $\mathcal{C}_T^I$ of $\mathcal{G}$
1 Construct the Dictionary $\mathcal{D}_e$;
2 $\mathcal{C}_T^I = \emptyset$;
3 **for** *Every* $(u, v) \in \mathcal{D}_e.keys()$ **do**
4 **if** $f_{(u,v)} \geq \gamma$ **then**
5 $\mathcal{T}_{(u,v)}$ = Time Stamps of (u, v);
6 $Temp = [\,]$;
7 $Temp.append(\mathcal{T}_{(u,v)}[1])$;
8 **for** $i = 2$ *to* $len(\mathcal{T}_{(u,v)})$ **do**
9 **if** $len(Temp) < \gamma$ **then**
10 **if** $\mathcal{T}_{(u,v)}[i] - Temp[1] \leq \Delta$ **then**
11 $Temp.append(\mathcal{T}_{(u,v)}[i])$;
12 **else**
13 $Temp = [\,]$;
14 $Temp.append$(time stamps of links occured in previous Δ $Duration$);
15 **end**
16 **else**
17 **if** $Temp[len(Temp) - \gamma + 1] + 1 + \Delta \geq \mathcal{T}_{(u,v)}[i]$ **then**
18 $Temp.append(\mathcal{T}_{(u,v)}[i])$;
19 **else**
20 $t_a = Temp[\gamma] - \Delta$; // first γ-th occurrence of (u, v) in *Temp*
21 $t_b = Temp[len(Temp) - \gamma + 1] + \Delta$; // last γ-th occurrence of (u, v) in *Temp*
22 $\mathcal{C}_T^I$.addClique($\{u, v\}, [t_a, t_b]$);
23 $Temp = [\,]$;
24 $Temp.append$(time stamps of the links occured in previous Δ $Duration$);
25 **end**
26 **end**
27 **if** $i = len(\mathcal{T}_{(u,v)})$ *and* $len(Temp) \geq \gamma$ **then**
28 $t_a = Temp[\gamma] - \Delta$; // first γ-th occurrence of (u, v) in *Temp*
29 $t_b = Temp[len(Temp) - \gamma + 1] + \Delta$; // last γ-th occurrence of (u, v) in *Temp*
30 $\mathcal{C}_T^I$.addClique($\{u, v\}, [t_a, t_b]$);
31 **end**
32 **end**
33 **end**
34 **end**

Lemma 1 *For a link* (u, v)*, if there exist any consecutive* γ *occurrences within* Δ *duration, then it has to be in* "*Temp*" *at some stage, in Algorithm* 1.

Proof Follows from the description of Algorithm 1. □

Lemma 2 *In any arbitrary iteration of the* "*for loop*" *at Line* 8 *in Algorithm* 1*, each consecutive* γ *occurrences of* "*Temp*" *will be within* Δ *duration.*

Proof Initially, *Temp* contains the first occurrence of a link. Now, when the length of *Temp* is less than γ (Line 9), next occurrence times are added in *Temp* (Line 11) if the difference from initial to current occurrence time lies within Δ (Line 10), else the times at which the links have occurred in previous Δ duration from the current time are added (Line 13, 14). This shows that all the entries in *Temp* are within Δ duration when the length of *Temp* is less than γ.

When the length of *Temp* is greater than or equal to γ, without loss of generality, let us take any arbitrary γ occurrences of *Temp* as $t^1, t^2, \ldots t^{(\gamma-1)}, t^\gamma$, which are not within a Δ duration, i.e., $t^\gamma - t^1 > \Delta$. Let us also assume that from $t^{(\gamma-1)}$, all the previous occurrences in *Temp* follow the statement of this lemma. Now, from our assumptions, we have the following conditions:

$$t^0 + \Delta \geq t^{\gamma-1} \implies t^1 + \Delta > t^{(\gamma-1)} \tag{1}$$

$$t^1 + \Delta < t^\gamma \tag{2}$$

$$t^1 \geq t^0 + 1 \tag{3}$$

Now, let us assume the previous occurrence of the link from t^1 in *Temp* is t^0, and our goal is to infer the possible positions of t^0 in the time horizon. From the definition of (Δ, γ)-clique, there will be γ occurrences from $t^1 - \Delta$ to t^1. If the first $(\gamma - 1)$ links have occurred in consecutive times, then $t^0 = t^1 - \Delta + \gamma - 2$. This is the minimum value for t^0. From Eq. 3, the maximum value for t^0 is $t^1 - 1$. Hence, $t^0 + 1 \leq t^1 \leq t^0 + \Delta + 2 - \gamma$. Now, from Eq. 2, we have $t^0 + \Delta + 1 < t^\gamma$, when $t^1 = t^0 + 1$ and replacing t^1 with $t^0 + \Delta + 2 - \gamma$ in Eq. 2, we get $t^0 + \Delta + 1 + (\Delta + 1 - \gamma) < t^\gamma \implies t^0 + \Delta + 1 < t^\gamma$ as $\Delta + 1 \geq \gamma$. This violates the condition imposed in Line 17. Hence, t^γ cannot be added in *Temp*. So, we reach a contradiction and this completes the proof. □

Lemma 3 *Let,* t^f *and* t^l *be the first and last occurrence in Temp. In the interval* $[t^f, t^l]$*, Temp contains at least* γ *links in each* Δ *duration.*

Proof When the length of *Temp* is less than γ, Lines 9–15 in Algorithm 1 ensure the statement of the lemma by adding consecutive γ occurrences in Δ duration. So, it is trivial that we need to prove the statement when the length of *Temp* is greater than γ. Let us assume that the occurrence times of the first $\gamma + 1$ entries of *Temp* are $t^1, t^2, \ldots, t^\gamma, t^{(\gamma+1)}$, where $t^1 = t^f$ and $t^{(\gamma+1)} \leq t^l$.

Now, by Lemma 2, $t^\gamma - t^1 \leq \Delta$ and $t^{(\gamma+1)} - t^2 \leq \Delta$. Without loss of generality, we want to show that there exist at least γ links from $t^1 + 1$ to $t^1 + 1 + \Delta$. As $t^\gamma - t^1 \leq \Delta$, the maximum difference between t^1 and t^2 can be $(\Delta - \gamma + 2)$, and this case will arise when all the $\gamma - 1$ links appear in each consecutive timestamp from $t^1 + \Delta$ toward t^1 (shown in Fig. 3). Now, as $t^{(\gamma+1)} - t^2 \leq \Delta$, we have to show $t^{(\gamma+1)} = t^\gamma + 1$. This extreme case will intuitively prove the rest of the cases. So, we can infer the following conclusion from Lemma 2 and the assumption $t^2 = t^1 + \Delta - \gamma + 2$. Now,

$$t^{(\gamma+1)} - t^2 \leq \Delta$$
$$t^{(\gamma+1)} - t^1 - \Delta + \gamma - 2 \leq \Delta$$
$$t^{(\gamma+1)} \leq t^1 + \Delta + 1 + \{(\Delta + 1) - \gamma\}$$

Again, from the condition imposed at Line 17 in Algorithm 1, we also have $t^{(\gamma+1)} \leq t^1 + \Delta + 1$. Now, as per our assumption of extreme case $t^\gamma = t^1 + \Delta$. So, $t^{(\gamma+1)} \leq t^\gamma + 1 \implies t^{(\gamma+1)} = t^\gamma + 1$.

Now, as $t^{(\gamma+1)} \leq t^1 + \Delta + 1$, we can argue $t^{(\gamma+1)} < t + \Delta$, for all $t \in (t^1 + 1, t^2]$. Moreover, from Lemma 2, there are γ links within $[t^2, t^{(\gamma+1)}]$, which concludes the existence of at least γ links from t to $t + \Delta$. Now, for any $t^i \in [t^f, t^l - \Delta]$, there will be at least γ links in *Temp* from t^i to $t^i + \Delta$. This completes the proof of the claimed statement. □

Lemma 4 *In Algorithm* 1*, the contents of* $\mathcal{C}_T^l$ *are* (Δ, γ) *-cliques of size* 2.

Proof We are processing each static edge of the temporal network $\mathcal{G}$ in its time horizon and add the (Δ, γ)-clique(s) formed by the end vertices of the edge into $\mathcal{C}_T^l$. Hence, the cliques in $\mathcal{C}_T^l$ are of size 2. Now, in Algorithm 1, the cliques are added into $\mathcal{C}_T^l$ in Lines 22 and 30. In both the cases, cliques are added if the current length of the *Temp* is greater than or equal to γ. As per Lemma 3, *Temp* at least γ links in each Δ duration. While adding the duration of the clique, t_a is obtained by subtracting Δ duration from first γ-th occurrence time, and t_b is obtained by adding Δ duration from last γ-th occurrence time in *Temp*. This ensures the existence of at least γ occurrences of the link in each Δ duration between t_a to t_b. □

Lemma 5 *All the cliques returned by Algorithm* 1 *and contained in* $\mathcal{C}_T^l$ *are duration-wise maximal.*

Proof We prove the duration-wise maximality of each clique in $\mathcal{C}_T^I$ by *contradiction*. Let us assume, a clique $(\{u, v\}, [t_a, t_b]) \in \mathcal{C}_T^I$ is not duration-wise maximal. Then, there exists a $t_a^{'}$ with $t_a^{'} < t_a$ such that $(\{u, v\}, [t_a^{'}, t_b])$ is a (Δ, γ)-clique or a $t_b^{'}$ with $t_b^{'} > t_b$ such that $(\{u, v\}, [t_a, t_b^{'}])$ is a (Δ, γ)-clique.

Now, if $(\{u, v\}, [t_a^{'}, t_b])$ is a (Δ, γ)-clique, then its first γ occurrences will be in *Temp* at some stage as per Lemma 1. Later, this *Temp* is expanded till t_b either by Line 11 or 18 in Algorithm 1. Hence, $(\{u, v\}, [t_a^{'}, t_b])$ will be added in $\mathcal{C}_T^I$, instead of $(\{u, v\}, [t_a, t_b])$. So, the assumption that there exists a $t_a^{'}$ with $t_a^{'} < t_a$ is false.

Now, by Lemma 4, as $(\{u, v\}, [t_a, t_b])$ is a (Δ, γ)-clique, in each Δ duration within t_a to t_b, there will be at least γ links between u and v. Let us assume, that l^{γ} and $l^{(\gamma-1)}$ are the last γ-th and $(\gamma - 1)$-th occurrence time of (u, v), respectively. From the definition of (Δ, γ)-clique, $l^{\gamma} + \Delta \geq t_b$, hence, $l^{(\gamma-1)} + \Delta > t_b$. Now, let $\{u, v\}$ be a (Δ, γ)-clique in the interval $[t_a, l^{(\gamma-1)} + \Delta]$, there must be at least one link between u and v in the interval $[t_b, l^{(\gamma-1)} + \Delta]$. If there exists such links, it indicates the presence of γ or more links in the interval $[l^{(\gamma-1)}, l^{(\gamma-1)} + \Delta]$. This case is handled by Algorithm 1 either in Line 11 or 18, and $(\{u, v\}, [t_a, t_b])$ will not be added to $\mathcal{C}_T^I$. So, there cannot exist any $t_b^{'}$ which is greater than t_b.

Hence, all the cliques of $\mathcal{C}_T^I$ returned by Algorithm 1 are duration-wise maximal. □

Lemma 6 *All the duration-wise maximal* (Δ, γ)*-cliques of size* 2 *are contained in* $\mathcal{C}_T^I$.

Proof In Lemmas 4 and 5, we have already shown that each (Δ, γ)-clique of $\mathcal{C}_T^I$ is of size 2 and duration-wise maximal, respectively. Hence, in this lemma, we have to prove that none of such cliques are missed out in the final $\mathcal{C}_T^I$. As each edge is processed independently by Algorithm 1, it is sufficient to prove that all the duration-wise maximal (Δ, γ) -cliques for a particular vertex pair (corresponding to an edge) are contained in $\mathcal{C}_T^I$.

Let, $(\{u, v\}, [t_a, t_b])$ be a duration-wise maximal (Δ, γ) -clique and not present in $\mathcal{C}_T^I$. Now, as $(\{u, v\}, [t_a, t_b])$ is a (Δ, γ)-clique, so there exist at least γ links in each Δ duration from t_a to t_b. Let f^{γ} and l^{γ} are the first γ-th and last γ -th occurrence time of the link (u, v) between t_a to t_b. We denote the occurrence timestamps for the static edge (u, v) as $t^1, t^2, \ldots, t^{f_{(u,v)}}$, and $f_{(u,v)} \geq \gamma$. Now, there can be one of the following cases for the values of t_a and t_b.

1. $t_a = t^{1+\gamma-1} - \Delta$ and $t_b \leq t^{f_{(u,v)}-\gamma+1} + \Delta$: The clique is formed at the beginning of the occurrence stream of (u, v). According to Lemma 1, all the occurrence time will be in *Temp*. Now, if $t_b = t^{f_{(u,v)}-\gamma+1} + \Delta$, it will be added in $\mathcal{C}_T^I$ by Line 30 of Algorithm 1. Otherwise, $\exists t^k : t^k > l^{\gamma} + 1 + \Delta$ and $t^{k-1} \leq t_b$. Hence, it breaks the if condition at Line 17, and the clique will be added in $\mathcal{C}_T^I$ by Line 22.
2. $t_a \geq t^{1+\gamma-1} - \Delta$ and $t_b = t^{f_{(u,v)}-\gamma+1} + \Delta$: The clique is formed at the end of the occurrence stream of (u, v). If $t_a = t^{1+\gamma-1} - \Delta$, it follows from the above case. For the else part, we need to show that $t_a = f^{\gamma} + \Delta > t^{1+\gamma-1} - \Delta$ is handled by the Algorithm 1. Here, $\exists t^k : t^k < f^{\gamma} - 1 - \Delta$ and $t^{k-1} \geq t_a$. Along with Lemma 1 and 2, the Lines 14 and 24 are responsible to have all the timestamps within $[t_a, t_b]$ must be *Temp*. So, the clique will be added in $\mathcal{C}_T^I$ by Line 30.
3. $t_a > t^{1+\gamma-1} - \Delta$ and $t_b < t^{f_{(u,v)}-\gamma+1} + \Delta$: The clique is formed in the middle of the occurrence stream of (u, v). Both the scenarios of t_a and t_b values are shown in the above two cases, so the clique will be added in $\mathcal{C}_T^I$ by Line 22.

□

Lemma 7 *The running time of finding all the duration-wise maximal* (Δ, γ)*-cliques of size* 2 *in Algorithm* 1 *is of* $\mathcal{O}(\gamma m)$.

Proof Preparing the dictionary $\mathcal{D}_e$ at Line 1 in Algorithm 1 will take $\mathcal{O}(\sum_{(u,v,t'') \in E(\mathcal{G})} f_{(u,v)})$. Assuming the frequency of each static edge is at least γ, we evaluate the running time for processing a static edge. It will be identical for the rest of the edges. During the processing, all the operations from Line 8 to 32 take $\mathcal{O}(1)$ times, except the appending at Lines 14 and 24. Now, the appending of previous occurrences within past Δ duration can lead to copying of at most $\gamma - 2$ previous entries in *Temp*, which take $\mathcal{O}(\gamma)$ times. Now, the worst case may occur when in every iteration of the for loop at Line 8, $\gamma - 2$ previous occurrences are copied in *Temp* (at Line 24), and this case may occur at most $f_{(u,v)} - \gamma + 1$ times. In this case, the running time of the for loop from Line 8 to 32 is $(\gamma - 2)(f_{(u,v)} - \gamma + 1) \approx \mathcal{O}(\gamma f_{(u,v)})$ for a particular static edge. Now, for all the static edges, the for loop at Line 3 will run with $\mathcal{O}(\sum_{(u,v,t'') \in E(\mathcal{G})} \gamma f_{(u,v)})$ times. Now, the total running time of Algorithm 1 is $\mathcal{O}(\sum_{(u,v,t'') \in E(\mathcal{G})} f_{(u,v)} + \gamma \sum_{(u,v,t'') \in E(\mathcal{G})} f_{(u,v)}) = \mathcal{O}(\gamma \sum_{(u,v,t'') \in E(\mathcal{G})} f_{(u,v)})$. Here, summing up all the frequencies of the static edges gives the total number of links of the temporal network, i.e., $m = \sum_{(u,v,t'') \in E(\mathcal{G})} f_{(u,v)}$. So, the time complexity of the initialization is of $\mathcal{O}(\gamma m)$. □

We have provided a weak upper bound on running time of the initialization process (Algorithm 1) in Lemma 7. Now, we focus on space requirement of Algorithm 1. Storing the Dictionary $\mathcal{D}_e$ in Line Number 1 requires $\mathcal{O}(m)$ space. In the worst case, space requirement by the list $\mathcal{T}_{uv}$ is of $\mathcal{O}(m)$. The size of *Temp* can go up to the maximum

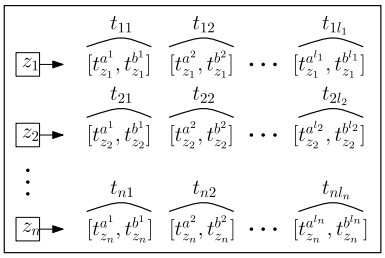

Fig. 4 The entries of $\mathcal{D}_{Temp}$ and $z_i \in \mathcal{D}_{Temp}.keys()$

number of times that any static edge has occurred consecutively more than gamma times in each delta duration, and in the worst case, it may take $\mathcal{O}(m)$ space. As all the initial cliques are of size 2, hence space requirement due to $\mathcal{C}_T^I$ is of $\mathcal{O}(n^2.f_{\max})$, where $f_{\max}$ is the highest frequency of the initial cliques. So, the total space requirement by Algorithm 1 is of $\mathcal{O}(m + n^2.f_{\max}) = \mathcal{O}(n^2.f_{\max})$. Hence, Lemma 8 holds.

Lemma 8 *The space requirement of Algorithm* 1 *is of* $\mathcal{O}(n^2.f_{\max})$.

Now for the temporal network shown in Fig. 1, the initial cliques with $\Delta = 3$ and $\gamma = 2$, in $\mathcal{C}_T^I$ are $(\{v_1, v_2\}, [1, 7])$, $(\{v_1, v_2\}, [7, 13])$, $(\{v_1, v_3\}, [2, 7])$, $(\{v_1, v_3\}, [8, 14])$, $(\{v_2, v_3\}, [2, 6])$, $(\{v_2, v_3\}, [7, 11])$, $(\{v_2, v_3\}, [5, 8])$, $(\{v_2, v_4\}, [4, 12])$, $(\{v_3, v_4\}, [1, 9])$, $(\{v_3, v_5\}, [5, 10])$, $(\{v_4, v_5\}, [4, 8])$.

4.2 Shrink and bulk phase (enumeration)

Algorithm 2 describes the enumeration strategy of our proposed methodology. For the given temporal network $\mathcal{G}$, we construct a static graph G where $V(G)$ is the vertex set of $\mathcal{G}$, and each link of $\mathcal{G}$ induces the corresponding edge in $E(G)$ without the time component, which we call as a static edge. Next, the dictionary $\mathcal{D}$ is built from the initial clique set $\mathcal{C}_T^I$ of Algorithm 1, where the vertex set of the clique is the key, and corresponding occurrence time intervals are the values. This data structure is also updated in the intermediate steps of Algorithm 2. Now, two sets $\mathcal{C}^{\mathcal{T}_1}$ and $\mathcal{C}^{\mathcal{T}_2}$ are maintained during the enumeration process. At any i-th iteration of the while loop at Line 5, $\mathcal{C}^{\mathcal{T}_1}$ maintains the current set of cliques which is yet to be processed for vertex addition and $\mathcal{C}^{\mathcal{T}_2}$ stores the new cliques formed in that i-th iteration. At the beginning, all the initial cliques from $\mathcal{C}_T^I$ are copied into $\mathcal{C}^{\mathcal{T}_1}$. A clique $(\mathcal{X}, [t_a, t_b])$ is taken out from $\mathcal{C}^{\mathcal{T}_1}$ which is duration-wise maximal, and the IS_MAX flag is set to TRUE for indicating the current clique as maximal (Δ, γ)-clique. For vertex addition, it is trivial to convince oneself that only for the neighboring vertices of $\mathcal{X}$ $(v \in \mathcal{N}_G(\mathcal{X}))$, there is a possibility of $(\mathcal{X} \cup \{v\}, [t_a^{'}, t_b^{'}])$ to be a (Δ, γ)-clique. If the new vertex set $\mathcal{X} \cup \{v\}$ is found in $\mathcal{D}$ with one of its value as $[t_a, t_b]$, the IS_MAX flag is set to FALSE, signifying that the processing clique $(\mathcal{X}, [t_a, t_b])$ is not maximal. Otherwise, if $\mathcal{X} \cup \{v\}$ is not present in $\mathcal{D}$, all the possible time intervals in which $\mathcal{X} \cup \{v\}$ can form a (Δ, γ)-clique are computed from Line 16 to 37. This process is iterated for all the neighboring vertices of $\mathcal{X}$ (Lines 10–38). Now, we describe the statements from Line 17 to 36 in detail. As mentioned earlier, to form a (Δ, γ)-clique with the new vertex set $\mathcal{X} \cup \{v\}$, all the possible combinations from $\mathcal{X} \cup \{v\}$ of size $|\mathcal{X}|$, (represented as $C(\mathcal{X} \cup \{v\}, \mathcal{X})$), have to be a (Δ, γ)-clique. Now, for all $z \in C(\mathcal{X} \cup \{v\}, \mathcal{X})$, if z is present in $\mathcal{D}.keys()$, it signifies the possibility of forming a new clique with the vertex set $\mathcal{X} \cup \{v\}$ (Line 17). Now, all the entries of these combinations are taken into a temporary data structure $\mathcal{D}_{Temp}$ from $\mathcal{D}$. For the clarity of presentation, we describe the operations from Line 19 to 35 for one vertex addition, i.e., $\mathcal{X} \cup \{v\}$ with the help of an example shown in Fig. 4. Now, let the entries of $\mathcal{D}_{Temp}$ be $z_1, z_2, \ldots z_n$, i.e., all $z_i \in C(\mathcal{X} \cup \{v\}, \mathcal{X})$, and the length of the corresponding entries in $\mathcal{D}_{Temp}$ be $l_1, l_2, \ldots l_n$, respectively. So, one sample from $z_1 \otimes z_2 \otimes \cdots \otimes z_n$ is taken as *timeSet* in Line 19 of Algorithm 2. One possible value of *timeSet* is $[t_{11}, t_{21}, \ldots, t_{n1}]$. For this value, the resultant interval $[t_a^{'}, t_b^{'}]$ is computed as $t_{11} \cap t_{21} \cdots \cap t_{n11} = [max(t_{z_1}^{a^1}, t_{z_2}^{a^1}, \ldots, t_{z_n}^{a^1}),\ min(t_{z_1}^{b^1}, t_{z_2}^{b^1}, \ldots, t_{z_n}^{b^1})]$. If the difference between $t_b^{'}$ and $t_a^{'}$ is more than or equal to Δ, then the newly formed (Δ, γ)-clique, $(\mathcal{X} \cup \{v\}, [t_a^{'}, t_b^{'}])$, is added in $\mathcal{C}^{\mathcal{T}_2}$and $\mathcal{D}$. Also, if $[t_a^{'}, t_b^{'}]$ matches with the current interval of $\mathcal{X}$, then the flag *IS_MAX* is set to FALSE, i.e., $(\mathcal{X}, [t_a, t_b])$ is not maximal. Now, this step is repeated for all the samples from $z_1 \otimes z_2 \otimes \cdots \otimes z_n$ from Line 19 to 35. This ensures that all the intervals in which $\mathcal{X} \cup \{v\}$ forms (Δ, γ) -clique are added in $\mathcal{D}$. Now, if none of the vertices from $\mathcal{N}_G(\mathcal{X}) \backslash \mathcal{X}$ is possible to add in $\mathcal{X}$, $(\mathcal{X}, [t_a, t_b])$ becomes a maximal (Δ, γ)-clique and added into final maximal clique set $\mathcal{C}_{\mathcal{L}}$ at Line 40. Vertex addition checking is performed for all the cliques of $\mathcal{C}^{\mathcal{T}_1}$ in the while loop from Line 7 to 42. When $\mathcal{C}^{\mathcal{T}_1}$ is exhausted and $\mathcal{C}^{\mathcal{T}_2}$ is not empty, the contents of $\mathcal{C}^{\mathcal{T}_2}$ are copied back into $\mathcal{C}^{\mathcal{T}_1}$ for further processing, signifying that all the maximal cliques have not been found yet. This is controlled using the flag *ALL_MAXIMAL* in the while loop at Line 5. If no clique is added into $\mathcal{C}^{\mathcal{T}_2}$, the flag *ALL_MAXIMAL* is set to TRUE so that in the next iteration, the condition of the while loop at Line 5 will be false, and finally, Algorithm 2 terminates. At the end, for the temporal network $\mathcal{G}$, $\mathcal{C}_T$ contains all the maximal (Δ, γ) -cliques of it. One illustrative example of the enumeration algorithm is given in Fig. 5.

Algorithm 2 Shrinking and bulking phase of the maximal (Δ, γ)-clique enumeration

Data: A Temporal Network $\mathcal{G}$, Initial Clique Set $\mathcal{C}_T^I$, Δ, γ.
Result: Maximal (Δ, γ) Clique Set $\mathcal{C}_T$ of $\mathcal{G}$.
1 Construct the Static Graph G;
2 Prepare the dictionary $\mathcal{D}$ from $\mathcal{C}_T^I$; // with the index as vertex set // and time intervals as entries
3 $\mathcal{C}^{\mathcal{T}_1} \leftarrow \mathcal{C}_T^I$;
4 ALL_MAXIMAL $= False$;
5 **while** $\neg$ *ALL_MAXIMAL* **do**
6 $\quad \mathcal{C}^{\mathcal{T}_2} \leftarrow \emptyset$;
7 $\quad$ **while** $\mathcal{C}^{\mathcal{T}_1} \neq \emptyset$ **do**
8 $\quad\quad$ Take and remove a clique $(\mathcal{X}, [t_a, t_b])$;
9 $\quad\quad$ IS_MAX $= True$;
10 $\quad\quad$ **for** *Every* $v \in \mathcal{N}_G(\mathcal{X}) \setminus \mathcal{X}$ **do**
11 $\quad\quad\quad \mathcal{X}_{new} = \mathcal{X} \cup \{v\}$;
12 $\quad\quad\quad$ **if** $\mathcal{X}_{new} \in \mathcal{D}$ **then**
13 $\quad\quad\quad\quad$ **if** $[t_a, t_b] \in \mathcal{D}[\mathcal{X}_{new}]$ **then**
14 $\quad\quad\quad\quad\quad$ IS_MAX $= False$;
15 $\quad\quad\quad\quad$ **end**
16 $\quad\quad\quad$ **else**
17 $\quad\quad\quad\quad$ **if** $\forall z \in \{C(\mathcal{X}_{new}, \mathcal{X})\}$ *and* $z \in \mathcal{D}$ **then**
18 $\quad\quad\quad\quad\quad \mathcal{D}_{Temp} \leftarrow$ Get the entries from $\mathcal{D}$ for C$(\mathcal{X}_{new}, \mathcal{X})$;
19 $\quad\quad\quad\quad\quad$ **foreach** *permutation of* $\mathcal{D}_{Temp}$ *entries as timeSet* **do**
20 $\quad\quad\quad\quad\quad\quad max\_t_a = [\,]$;
21 $\quad\quad\quad\quad\quad\quad min\_t_b = [\,]$;
22 $\quad\quad\quad\quad\quad\quad$ **for** $t \in timeSet$ **do**
23 $\quad\quad\quad\quad\quad\quad\quad max\_t_a.append(t[1])$;
24 $\quad\quad\quad\quad\quad\quad\quad min\_t_b.append(t[2])$;
25 $\quad\quad\quad\quad\quad\quad$ **end**
26 $\quad\quad\quad\quad\quad\quad t_a^{'} = MAX(max\_t_a)$;
27 $\quad\quad\quad\quad\quad\quad t_b^{'} = MIN(min\_t_b)$;
28 $\quad\quad\quad\quad\quad\quad$ **if** $t_b^{'} - t_a^{'} \geq \Delta$ **then**
29 $\quad\quad\quad\quad\quad\quad\quad \mathcal{C}^{\mathcal{T}_2}.add(\mathcal{X}_{new}, [t_a^{'}, t_b^{'}])$;
30 $\quad\quad\quad\quad\quad\quad\quad \mathcal{D}[\mathcal{X}_{new}].append([t_a^{'}, t_b^{'}])$;
31 $\quad\quad\quad\quad\quad\quad\quad$ **if** $t_a^{'} = t_a \wedge t_b^{'} = t_b$ **then**
32 $\quad\quad\quad\quad\quad\quad\quad\quad$ IS_MAX $= False$;
33 $\quad\quad\quad\quad\quad\quad\quad$ **end**
34 $\quad\quad\quad\quad\quad\quad$ **end**
35 $\quad\quad\quad\quad\quad$ **end**
36 $\quad\quad\quad\quad$ **end**
37 $\quad\quad\quad$ **end**
38 $\quad\quad$ **end**
39 $\quad\quad$ **if** *IS_MAX* **then**
40 $\quad\quad\quad \mathcal{C}_T.append(\mathcal{X}, [t_a, t_b])$;
41 $\quad\quad$ **end**
42 $\quad$ **end**
43 $\quad$ **if** $len(\mathcal{C}^{\mathcal{T}_2}) > 0$ **then**
44 $\quad\quad \mathcal{C}^{\mathcal{T}_1} \leftarrow \mathcal{C}^{\mathcal{T}_2}$;
45 $\quad$ **else**
46 $\quad\quad$ ALL_MAXIMAL $= True$;
47 $\quad$ **end**
48 **end**

Now, from the description of the enumeration process of our proposed methodology, we have the following claims:

Claim 1 *For any arbitrary clique* $(\mathcal{X}, [t_a, t_b]) \in \mathcal{C}^{\mathcal{T}_1}$ *and* $v \in \mathcal{N}_G(\mathcal{X}) \backslash \mathcal{X}$, *all the time intervals in the whole lifespan of the linked stream* $\mathcal{L}$, *at which* $\mathcal{X} \cup \{v\}$ *forms a* (Δ, γ)*-clique, are added in* $\mathcal{D}$.

Claim 2 *In any arbitrary iteration* i *of the while loop at Line* 5, *the cliques of* $\mathcal{C}^{\mathcal{T}_1}$ *and* $\mathcal{C}^{\mathcal{T}_2}$ *are of size* $i + 1$ *and* $i + 2$, *respectively*.

Lemma 9 *In Algorithm* 2, *the elements of* $\mathcal{C}_T$ *are* (Δ, γ)*-cliques*.

Proof All the cliques are added in $\mathcal{C}_T$, only from $\mathcal{C}^{\mathcal{T}_1}$ at Line 40 in Algorithm 2. Now, initially $\mathcal{C}^{\mathcal{T}_1}$ contains the elements from $\mathcal{C}_T^I$, which are (Δ, γ)-cliques from Lemma 4, and later, it is updated with the entries of $\mathcal{C}^{\mathcal{T}_2}$. So, if we show that the elements of $\mathcal{C}^{\mathcal{T}_2}$ are (Δ, γ)-cliques, the statement will be proved. Now, all the cliques of $\mathcal{C}^{\mathcal{T}_2}$ are of at least Δ duration, from the condition at Line 28. Also, from the description of the Algorithm 2, it is easy to verify that in each iteration of

vertex, addition to a clique of $\mathcal{C}^{\mathcal{T}_1}$ can only be made, if all the possible combinations of vertices form (Δ, γ)-cliques. This ensures that all the vertex pairs of the clique in $\mathcal{C}^{\mathcal{T}_2}$ are linked at least γ times in each Δ duration within the intersected time interval of all the combinations. Hence, the elements of $\mathcal{C}_T$ are (Δ, γ)-cliques.

□

Lemma 10 *In Algorithm* 2, *all the intermediate cliques are duration-wise maximal.*

Proof From the proof of Lemma 9, it is sufficient to show that the contents of $\mathcal{C}^{\mathcal{T}_1}$ are duration-wise maximal. We prove the statement by induction. From Lemma 5, the contents of initial clique set are duration-wise maximal. Let us assume that in the i-th iteration of the while loop at Line 5, the contents of $\mathcal{C}^{\mathcal{T}_1}$ are duration-wise maximal. We need to show that the same will hold in the $(i+1)$-th iteration also. After adding a vertex to an existing clique obtained in i-th iteration for possible expansion, the new vertex set is considered to be a (Δ, γ)-clique within the intersected interval of all $(i+2)$-combinations, if the length of the intersected interval is more than Δ (Lines 17–36 in Algorithm 2). Now, it can be observed that the latest first γ-th occurrence time (f^{γ}_{i+1}) of the resultant clique must be same with the latest first γ-th occurrence time (f^{γ}_{i}) of the constituting clique from which t_a is coming. Similarly, the earliest last γ-th occurrence time (l^{γ}_{i+1}) of the resultant clique must be same with the earliest last γ-th occurrence time (l^{γ}_{i}) of the constituting clique from which t_b is coming. When both the t_a and t_b are coming from the same constituting clique, the original clique is not maximal as vertex addition is possible. Now, for the resultant clique, the beginning time t_a can not be extended to $t_a - 1$ as in the i-th iteration, the constituting clique is also duration-wise maximal from the assumption, i.e., $f^{\gamma}_{i} - \Delta = t_a \implies f^{\gamma}_{i+1} - \Delta = t_a$. Similarly, t_b can not be extended to $t_b + 1$ as in the i-th iteration, the constituting clique is also duration-wise maximal from the assumption, i.e., $l^{\gamma}_{i} + \Delta = t_b \implies l^{\gamma}_{i+1} + \Delta = t_b$. So, the resultant clique at $(i+1)$-th iteration is also duration-wise maximal. This is true for all the cliques generated in each iteration. Hence, all the intermediate cliques in Algorithm 2 are duration-wise maximal. □

Lemma 11 *In Algorithm* 2, *at the begining of any i-th iteration,* $\mathcal{C}^{\mathcal{T}_1}$ *holds all the duration-wise maximal* (Δ, γ)*-cliques of size* $i+1$.

Proof For $i = 1$, $\mathcal{C}^{\mathcal{T}_1}$ holds all the duration-wise maximal (Δ, γ)-cliques of size 2 from Lemma 6. Let, $\mathcal{C}^{\mathcal{T}_1}_{i-1}$ and $\mathcal{C}^{\mathcal{T}_1}_{i}$ are the clique sets at the beginning of the iteration $i-1$ and i, respectively, and $\mathcal{C}^{\mathcal{T}_1}_{i-1}$ holds all the duration-wise maximal (Δ, γ)-cliques of size i. Then, we have to show that during the construction of $\mathcal{C}^{\mathcal{T}_1}_{i}$ from $\mathcal{C}^{\mathcal{T}_1}_{i-1}$, the clique set $\mathcal{C}^{\mathcal{T}_1}_{i}$ remains exhaustive. For a clique from $\mathcal{C}^{\mathcal{T}_1}_{i-1}$, we check for all the possible $i+1$ vertex combinations in Line 17 of Algorithm 2, which does not leave any possible vertex addition to the clique. Next, for each added vertex, all the possible time interval combinations are generated and checked from Line 19 to 35. Now, for each possible time combination, the (Δ, γ) -clique is generated from the maximum possible common interval of them. This guarantees that all the possible cliques are generated during this process. Again, from Lemma 10, in the i-th iteration, all the generated cliques are also duration-wise maximal, which are now in $\mathcal{C}^{\mathcal{T}_1}_{i}$. So, the same can be proved in the clique building from i-th to $i+1$-th iteration. Hence, for any value of i, the claimed statement is true.

□

Lemma 12 *All the* (Δ, γ)*-cliques returned by Algorithm* 2 *and contained in* $\mathcal{C}_T$ *are maximal.*

Proof We prove this statement by contradiction. Assume that $C_i = (\mathcal{X}, [t_a, t_b])$ be an element of $\mathcal{C}^{I}_{T}$, which is not maximal. In Algorithm 2, the cliques are added in $\mathcal{C}^{I}_{T}$ from $\mathcal{C}^{\mathcal{T}_1}$, and all the cliques in $\mathcal{C}^{\mathcal{T}_1}$ are duration-wise maximal (Δ, γ) -cliques from Lemma 10. If, C_i is not maximal, then the only thing that can happen is that one or more vertex addition is possible to make C_i maximal. Now, let us assume that $\exists v \in \mathcal{N}_G(\mathcal{X})$, such that $(\mathcal{X} \cup \{v\}, [t_a, t_b])$ is a (Δ, γ)-clique. From the enumaration process described in Algorithm 2, if a clique is added to $\mathcal{C}_{\mathcal{L}}$, it has to be in $\mathcal{C}^{\mathcal{T}_1}$ in any previous iteration. As $(\mathcal{X} \cup \{v\}, [t_a, t_b])$ is a (Δ, γ)-clique, the *IS_MAX* flag becomes FALSE so that it is not going to be added in $\mathcal{C}_{\mathcal{L}}$ but in $\mathcal{C}^{\mathcal{T}_2}$. Hence, the assumption $C_i \in \mathcal{C}_{\mathcal{L}}$ is a contradiction. So, all the elements of $\mathcal{C}_{\mathcal{L}}$ returned by Algorithm 2 are maximal (Δ, γ)-cliques. □

Theorem 1 *All the maximal* (Δ, γ)*-cliques of* $\mathcal{G}$ *are contained in* $\mathcal{C}_T$.

As mentioned previously, m denotes the temporal links in the time-varying graph $\mathcal{G}$. At Line Number 2, computing the static graph from the given time-varying graph requires $\mathcal{O}(m)$ time. Time requirement for creating the dictionary $\mathcal{D}$ will be of $\mathcal{O}(|\mathcal{C}_T|.f_{\}}$ time, where $f_{\max}$ denotes the highest number of times a clique appeared. Copying the cliques from the list $\mathcal{O}(\mathcal{C}_T)$ to $\mathcal{C}^{T_1}$ requires $\mathcal{O}(|\mathcal{C}_T|)$ time. Setting the *ALL_MAXIMAL* flag to FALSE in Line Number 4 requires $\mathcal{O}(1)$ time. So, from Line Number 1 to 4, the time requirement is of $\mathcal{O}(m + |\mathcal{C}^{I}_{T}|.f_{\}}$. Now, it is easy to verify that the instructions in Line Numbers 6, 8, and 9 require $\mathcal{O}(1)$ time. The `for` loop in Line Number 10 can run at most $\mathcal{O}(n)$ time. Adding the vertex v to the existing clique $\mathcal{X}$ to form

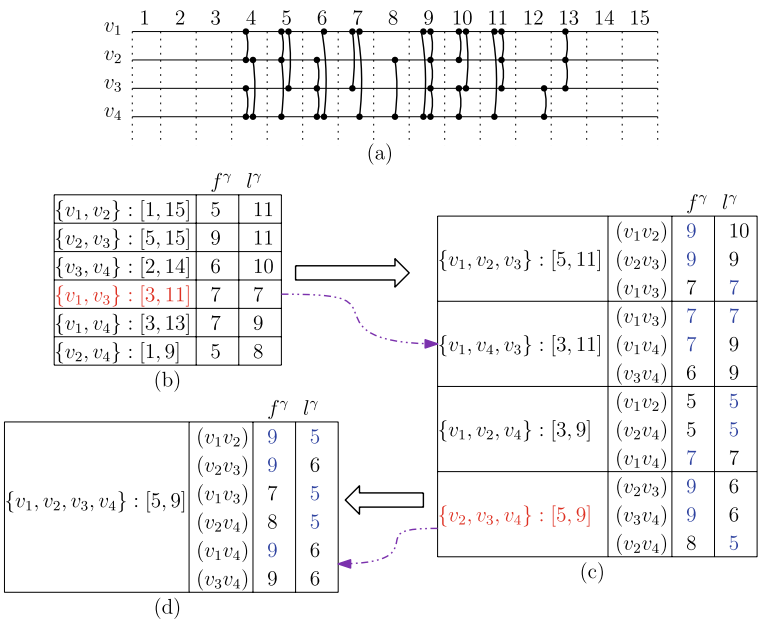

Fig. 5 Illustrative example of the proposed maximal (Δ, γ) -clique enumeration algorithm, **a** input temporal graph with $\Delta = 4$ and $\gamma = 2$, **b** output of the Algorithm 1—stretching phase, and **c** and **d** the content of $\mathcal{C}^{\mathcal{T}_1}$ at different iterations of Algorithm 2. The cliques in red are duration-wise maximal but not w.r.t. cardinality

$\mathcal{X}_{new}$ in Line Number 11 requires $\mathcal{O}(1)$ time. The maximum number of comparisons in the condition of the `if` statement in Line Number 12 will be $\mathcal{O}(|\mathcal{C}_T|)$. In the worst case, each comparison can take at most $\mathcal{O}(n^2)$ time. Hence, the total time requirement for Line Number 12 requires $\mathcal{O}(|\mathcal{C}_T|.n^2)$ time. Number of comparisons in the conditional statement in Line Number 13 requires at most $\mathcal{O}(f_)$ time. Setting the *IS_MAX* flag to "False" in Line Number 14 requires $\mathcal{O}(1)$ time. Now, in the `if` statement of Line Number 17, the number of combinations can be $\mathcal{O}(n)$ in the worst case. Hence, the number of comparisons for checking the existence in the dictionary $\mathcal{D}$ is of $\mathcal{O}(n|\mathcal{C}_T|)$. As mentioned previously, each individual comparison requires $\mathcal{O}(n^2)$ time. Hence, total execution time for Line 17 is of $\mathcal{O}(n^3.|\mathcal{C}_T|)$ time. Now, copying the newly generated combinations from the dictionary $\mathcal{D}$ to $\mathcal{D}_{Temp}$ requires $\mathcal{O}(nf_{max})$. It can be verified from the description of the Algorithm 2 that the number of possible combinations among the time duration is of $\mathcal{O}(f^n_{max})$. Hence, the for loop in Line Number 19 will execute $\mathcal{O}(f^n_{max})$ times. Line Numbers 20 and 21 take $\mathcal{O}(1)$ time. Executing the for loop from Line Number 22 to 25 requires $\mathcal{O}(n)$ time. Computing the maximum and minimum value among the elements of the list max_t_a and min_t_b requires $\mathcal{O}(n)$ time. It is easy to verify that execution of Line Number 28 to 34, 39 to 41, 43 to 45 and 46 requires $\mathcal{O}(1)$ time. Copying the cliques from in Line Number 44 can take $\mathcal{O}(|\mathcal{C}_T|)$ time. Now, we need to wrap up the computational time requirement for the looping structures to obtain the total time requirement of Algorithm 2. From the previous analysis, it can be verified that the time requirement for executing the for loop from Line Number 19 to 35 will be of $\mathcal{O}(f^n_{max}.n)$. The for loop from Line Number 10 to 38 will execute at max $\mathcal{O}(n)$ times. Hence, the running time from 10 to 38 is of $\mathcal{O}(n(n^2.|\mathcal{C}_T|.f_{max} + n^3.|\mathcal{C}_T| + n.f_{max} + f^n_{max}.n)) = \mathcal{O}(n^3.|\mathcal{C}_T|.f_{max} + n^4.|\mathcal{C}_T| + n^2.f_{max} + f^n_{max}.n^2) = \mathcal{O}(n^3.|\mathcal{C}_T|.f_{max} + n^4.|\mathcal{C}_T| + f^n_{max}.n^2)$. The while loop from Line Number 7 to 42 can execute at most $\mathcal{O}(|\mathcal{C}_T|)$ times. Hence, execution time of this while loop is of $\mathcal{O}(n^3.|\mathcal{C}_T|^2.f_{max} + n^4.|\mathcal{C}_T|^2 + |\mathcal{C}_T|.f^n_{max}.n^2)$. Also, the number of times the while loop from Line Number 5 to 48 can execute is at most $\mathcal{O}(n)$ times. Hence, time requirement for execution of Line Numbers 5–48 is $\mathcal{O}(n(n^3.|\mathcal{C}_T|^2.f_{max} + n^4.|\mathcal{C}_T|^2 + |\mathcal{C}_T|.f^n_{max}.n^2 + |\mathcal{C}_T|)) = \mathcal{O}(n^4.|\mathcal{C}_T|^2.f_{max} + n^5.|\mathcal{C}_T|^2 + |\mathcal{C}_T|.f^n_{max}.n^3 + n.|\mathcal{C}_T|) = \mathcal{O}(n^4.|\mathcal{C}_T|^2.f_{max} + n^5.|\mathcal{C}_T|^2 + |\mathcal{C}_T|.f^n_{max}.n^3)$. As already derived that running time from Line Number 1 to 4 is of $\mathcal{O}(m + |\mathcal{C}^I_T|.f_{max})$, hence, the total time requirement for Algorithm 2 is of $\mathcal{O}(n^4.|\mathcal{C}_T|^2.f_{max} + n^5.|\mathcal{C}_T|^2 + |\mathcal{C}_T|.f^n_{max}.n^3 + m + |\mathcal{C}^I_T|.f_{max}) = \mathcal{O}(n^4.|\mathcal{C}_T|^2.f_{max} + n^5.|\mathcal{C}_T|^2 + |\mathcal{C}_T|.f^n_{max}.n^3)$. Maximum number of cliques could be at $\{\{\{\text{max}\}\}\}\max 2^n$. Hence, plugging the worst case value of $|\mathcal{C}_T|$, we have the running time of Algorithm 2, is $\mathcal{O}(n^4.2^{2n}.f_{\max} + n^5.2^{2n} + 2^n.f^n_{\max}.n^3)$.

Additional space requirement of the Algorithm 2 is due to the "static graph" G, which requires $\mathcal{O}(m)$ space; dictionary $\mathcal{D}$, which requires $\mathcal{O}(|\mathcal{C}^I_T|.f_{\max})$ space; dictionary $\mathcal{D}_{Temp}$ which requires $\mathcal{O}(n.f_{\max})$ space, the list $\mathcal{X}_{new}$ which requires $\mathcal{O}(n)$ space, the lists $\mathcal{C}^{\mathcal{T}_1}$, $\mathcal{C}^{\mathcal{T}_2}$, and $\mathcal{C}_T$ which in the worst case these may require $\mathcal{O}(n2^n)$ space; and the lists max_t_a and min_t_b which require $\mathcal{O}(|\mathcal{C}_T|)$ space. Hence, total space requirement of Algorithm 2 is of $\mathcal{O}(m + |\mathcal{C}^I_T|.f_{\max} + n.f_{\max} + n + n.2^n + 2^n) = \mathcal{O}(m + |\mathcal{C}^I_T|.f_{\max} + n.f_{\max} + n.2^n)$. Hence, Lemma 13 holds.

Lemma 13 *The running time and space requirement of Algorithm 2 are of* $\mathcal{O}(n^4 \cdot 2^{2n} \cdot f_{\max} + n^5 \cdot 2^{2n} + 2^n \cdot f^n_{\max} \cdot n^3)$ *and* $\mathcal{O}(m + |\mathcal{C}^I_T| \cdot f_{\max} + n \cdot f_{\max} + n \cdot 2^n)$, *respectively.*

As mentioned previously, Algorithm 1 and 2 together constitute the proposed enumeration strategy for maximal (Δ, γ)-cliques of a temporal network. It has been shown in Lemma 7 that the time requirement of Algorithm 1 is of $\mathcal{O}(\gamma.m)$. Hence, the total time requirement of the proposed methodology (i.e., Algorithm 1 and 2) is of $\mathcal{O}(n^4.2^{2n}.f_{\max} + n^5.2^{2n} + 2^n.f_{\max}^n.n^3 + \gamma.m)$. As mentioned in Lemma 8, the space requirement is of $\mathcal{O}(n^2.f_{\max})$. Hence, total space requirement of the proposed methodology is of $\mathcal{O}(m + |\mathcal{C}_T^I|.f_{\max} + n.f_{\max} + n.2^n + n^2.f_{\max}) = \mathcal{O}(m$ $\mathcal{O}(m + |\mathcal{C}_T^I|.f_{\max} + n.f_{\max} + n.2^n + n^2.f_{\max}) = \mathcal{O}(m$. Now, the Theorem 2 states regarding the time and space requirement of the proposed methodology.

Theorem 2 *The computational time and space requirement of the proposed methodology are of* $\mathcal{O}(n^4 \cdot 2^{2n} \cdot f_{\max} + n^5 \cdot 2^{2n} + 2^n \cdot f_{\max}^n \cdot n^3 + \gamma \cdot m)$ *and* $\mathcal{O}(m + |\mathcal{C}_T^I| \cdot f_{\max} + n \cdot 2^n + n^2 \cdot f_{\max})$, *respectively.*

5 Experimental evaluation

In this section, we present the experimental evaluation of the proposed methodology and compare its efficacy with the existing methods from the literature. First, we outline the background of the used datasets, followed by the objectives, comparing algorithm description, and discussion on results.

5.1 Description of the datasets

In our experiments, we have used the following datasets: (1) Hypertext 2009 dynamic contact network (Hypertext) (Isella et al. 2011): This dataset was collected during the ACM Hypertext 2009 conference, where the attendees volunteered to wear radio badges that monitored their face-to-face proximity. The dataset represents the dynamical network of face-to-face proximity of 110 conference attendees over about 2.5 days. (2) College Message Temporal Network (College Message) (Panzarasa et al. 2009): This dataset contains the interaction information among a group of students from the University of California, Irvine. (3) Bitcoin OTC Trust Weighted Signed Network (Bitcoin)[1] (Kumar et al. 2016, 2018): This is a who-trusts-whom network of people who trade using Bitcoin on a platform called *Bitcoin OTC*. Members of Bitcoin OTC rate other members on a scale of -10 (total distrust) to +10 (total trust) in steps of 1. This is a weighted, signed, and directed network. However, as per our requirement, we do not consider the direction and weight. As trust of a person changes over time, it is a temporal network. (4) Infectious SocioPatterns Dynamic Contact Network I & II (Infectious I & II) (Isella et al. 2011): The datasets are collected during the Infectious SocioPatterns event that took place in Dublin, Ireland, during the art science exhibition INFECTIOUS: STAY AWAY. The dataset contains the set of tuples of the form (t, u, v), where u and v are the anonymous ids of the person who are in contact for at least 20 s. Basic statistics of the datasets are given in Table 2.

Table 2 Basic statistics of the datasets

Datasets	#Nodes	#Links	#Static edges	Lifetime
Hypertext	113	20,818	2196	2.5 days
Infectious II	410	17,298	2765	8 h
College message	1899	59,835	20,296	193 days
Bitcoin	5881	35,592	21,492	5.21 years
Infectious I	10,972	415,843	44,516	80 days

5.2 Setup of the experimentation

The only parameters involved in our study are Δ and γ. For analyzing a temporal network dataset, one intuitive question will be just to find out the frequently connected groups for a given time duration, which is comparable with the lifetime of the network. For this reason, we select the Δ value based on the network lifetime only. For "Hypertext" and "Infectious II" datasets, we start with the Δ value of 1 min and keep on increasing it by 1 min till it reaches 10 min. Whereas it is increased in multiplicative order of 10 starting from 1 and 2 min to 100 and 200 min in the "Infectious I" dataset, due to its larger lifetime. For "College Message" and "Bitcoin" datasets, we choose the Δ value as 1, 12, 64, 72, and 168 h.

For Δ-clique enumeration in all the datasets, we have to set γ value as 1. Now, for enumerating (Δ, γ)-clique, in case of the "Hypertext" and "Infectious II," we start with the γ value as 2, keep on increasing it by 1 till the maximal clique set becomes empty. In case of "Infectious I" dataset for initial Δ values (e.g., 60, 120), we start that γ value is chosen similarly with that of the "Infectious II" dataset. However, for larger Δ values (e.g., 6000, 12000), we start with a γ value of 5, and then 10; next incremented by 10 till it reaches 30, and subsequently incremented by 30 till it reaches 330. For the "Bitcoin" dataset, for every Δ value, if we increase the γ value beyond 2, the maximal clique set becomes null, due to very small links per static edges ratio, compared to the lifespan of the temporal network. In case of "College Message" dataset, as the chosen Δ value is larger, the γ value is incremented by 5 till it goes to 20 and then by 10 till the maximal clique set becomes empty. The goals of the experiments are to analyze, how the number of maximal

[1] https://snap.stanford.edu/data/soc-sign-bitcoin-otc.html.

cliques, maximum cardinality, maximum duration, computational time, and space change with Δ and γ, and compare the results with the existing algorithms.

5.3 Algorithms compared

In our experiments, we compare the performance of the proposed methodology with the following methods from the literature. (1) Virad et al.'s method (Viard et al. 2016): This is the first method proposed to enumerate maximal Δ-cliques of a temporal network. (2) Himmel et al.'s method (Himmel et al. 2017): This method incorporates the famous Born–Kerbosch algorithm to improve Virad et al.'s method. (3) Banerjee & Pal's method (Banerjee and Pal 2019): This is the existing maximal (Δ, γ)-clique proposed by us in one of our previous studies. We obtain the source code of the first two methodologies as implemented by the respective authors. The proposed methodology is developed in Python 3.4 along with NetworkX 2.0. All the experiments have been carried out on a high-performance computing cluster having 5 nodes, and each of them having 40 cores and 160 GB of RAM. Implementations of the algorithms are available at https://github.com/BITHIKA1992/Delta-gamma-Clique-Update.

5.4 Experimental results with discussions

Results for Δ-clique enumeration: First, we focus on Δ -clique, which is equivalent to (Δ, γ)-clique with $\gamma = 1$. This result is shown in Table 3. In all the datasets, the maximal clique count(N) decreases with the increment of Δ. This quantity increases for a large Δ when there exist some user pairs who contact each other very frequently for a long duration. This generates many maximal cliques with cardinality 2

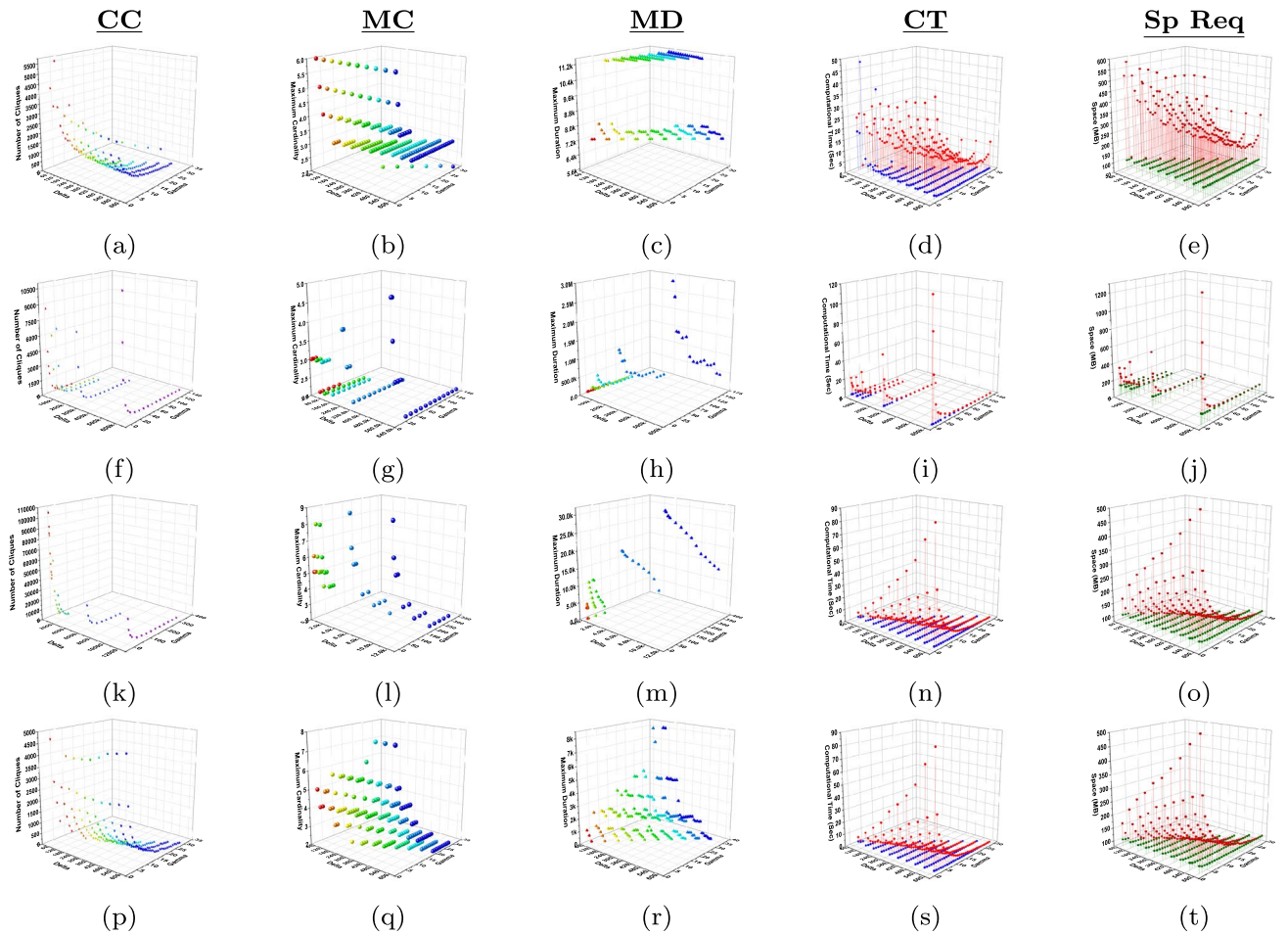

Fig. 6 Plots for the change in Clique Count (denoted as CC), Maximum Cardinality (denoted as MC), Maximum Duration (denoted as MD), Computational Time (denoted as CT), and Space Requirement (denoted as SR) with the change of Δ and γ for different datasets; **a**–**e** Hypertext; **f**–**j** College Message; **k**–**o** Infectious I; and **p**–**t** Infectious II; the computational time and space of Banerjee & Pal's (Banerjee and Pal 2019) marked in red

Table 3 Results for the Maximal Clique Count (N), Maximum Cardinality (C), Maximum Duration (D), Computational Time (in Secs.) / Space (in MB) for maximal Δ-clique ((Δ, γ)-clique with $\gamma = 1$) enumeration for different datasets

Dataset	Δ	*N*	*C*	*D*	Algorithm		
					Viard et al. (2016)	Himmel et al. (2017)	Proposed
Hypertext	60	7897	7	7640	16.02 / 208	6.14 / 104	53.9 / 109
	120	6859	7	8140	18.11 / 221	4.48 / 103	26.14 / 108.42
	180	6453	7	11,520	20.17 / 237	3.8 / 103	16.35 / 108.37
	240	6232	7	11,640	21.73 / 247	3.73 / 103	11.76 / 108.27
	300	6106	7	11,760	23.11 / 256	3.62 / 103	9.73 / 108.4
	360	6025	7	11,880	24.02 / 268	3.31 / 102.85	8.55 / 108.49
	420	5980	7	12,000	25.54 / 281	3.41 / 102.8	7.76 / 108.47
	480	5952	7	12,120	26.61 / 291	3.28 / 102.78	7.27 / 108.54
	540	5930	7	17,600	28.61 / 308	3.23 / 102.74	6.6 / 108.61
	600	5913	7	17,720	29.83 / 318	3.08 / 102.72	6.12 / 108.68
College message	3600	33,933	4	21,761	35.25 / 372	41 / 140	19.84 / 148
	43,200	25,635	5	403,018	43.02 / 546	31.38 / 133	4.19 / 142
	88,640	22,701	5	896,134	52.29 / 727	28.56 / 131	2.28 / 140
	259,200	21,019	5	2,322,612	84.05 / 1281	27.61 / 128	1.41 / 136
	604,800	21,658	6	6,334,253	133.53 / 2427	25.85 / 128	1.19 / 139
Bitcoin	3600	26,577	7	10,791	18.31 / 142.97	196.49 / 221	2.36 / 157
	43,200	26,091	8	129,422	19.58 / 142.71	193.74 / 235	2.41 / 158
	88,640	25,970	8	265,798	20.69 / 142.57	190.93 / 250	2.3 / 159
	259,200	26,290	8	777,572	22.6 / 142.73	191.49 / 288	2.36 / 160
	604,800	27,149	8	1,814,344	29.69 / 143.39	193.52 / 367	2.34 / 162
Infectious I	60	161,066	6	3760	274.13 / 2591	1025.01 / 286	80.75 / 357
	120	138,662	7	5180	405.84 / 3915	998.53 / 266	46.88 / 354
	600	128,392	10	11,200	2659.82 / 18117	1043.46 / 262	30.87 / 523
	1200	139,684	13	12,400	9824.58 / 72628	1062.69 / 278	84.13 / 1017
	6000	152,121	16	22,740	NA	1238.75 / 293	108.34 / 3076
	12,000	152,198	16	34,740	NA	1266.8 / 293	108.03 / 3097
Infectious II	60	9776	5	1860	10.77 / 197	4.71 / 106	4.41 / 111.1
	120	9397	6	2900	17.36 / 266	4.12 / 105.8	2.39 / 112.25
	180	9565	7	4280	26.41 / 339	4.04 / 105.82	1.7 / 113.76
	240	9849	7	5160	37.74 / 429	3.97 / 106	1.81 / 115.39
	300	10,192	8	5280	51.8 / 540	4.13 / 106	1.76 / 117.7
	360	10,734	8	6480	68.27 / 661	4.34 / 106.65	1.88 / 120.96
	420	11,287	8	6600	92.06 / 815	4.68 / 107.14	2.21 / 124.43
	480	11,571	9	8540	122.26 / 1013	5.12 / 107.52	2.85 / 130
	540	11,781	9	9580	159.73 / 1245	5.45 / 107.77	2.75 / 133.17
	600	12,123	10	9700	201.34 / 1498	5.96 / 108.17	3.5 / 137.44

and different $[t_a, t_b]$ for bitcoin and infectious. The maximum cardinality(C) identifies if there is a large group, and the maximum duration(D) signifies maximum how long users contacted each other. Both C and D are non-decreasing with the growth in Δ, in all the datasets except "Hypertext." It is observed that the proposed methodology is the fastest one compared to the existing methods. The computational time increases with Δ in Viard et al. (2016), as the algorithm starts with the clique(link) with duration=1, and expands in both right and left by Δ and creates more intermediate cliques. Whereas, the processing time depends on the maximal clique count in both (Himmel et al. 2017) and the proposed method. The computational space is mainly dependent on the size of the intermediate clique set, and it gets penalized more in Virad et al.'s method due to the same reason as discussed. The effect can be seen in "Infectious I" for $\Delta = 6000$ and 12000. The system's memory becomes insufficient to compute for these two Δ values. For all the datasets, the proposed method beats Virad et al.'s method, both in terms of space and time. Comparing with

the Himmel et al.'s method with the proposed method, the trade-off between time and space can be observed for the dense datasets.

Results for (Δ, γ)-clique enumeration: The results for $\gamma > 1$ are shown in Fig. 6. As the maximal clique set becomes null for $\gamma > 2$ in Bitcoin, we do not show the plots for it. For a fixed Δ, the maximal clique count decreases exponentially with the increase in γ (refer to Fig. 6 [a,f,k,p]), which, in turn, reduces the computational time and space as well. Maximum cardinality and the duration also reduce with the increment in γ. For fixed γ, the same observation of $\gamma = 1$ is found. While comparing with the only existing method (Banerjee and Pal 2019), it can be observed that the improvement is more significant for larger value of Δ and the small value of γ (Refer to Fig. 6 [d,e,i,j,n,o,s,t]). Lastly, we can conclude for both Δ and (Δ, γ)-clique enumeration, the proposed methodology is better when the input dataset is sparse. Among all the datasets, hypertext and Infectious II are comparatively more dense than others, in terms of static graph density and number of links per timestamp. Hypertext dataset is the most dense network with density 34.7% (3.2% for Infectious II), and a node in the hypertext dataset is involved in three links at a time (5 for Infectious II). The same for other datasets are in the order of $\sim 10^{-3}$, which are much lower. Now, from Table 3, it can be seen that Himmel et al. perform much better than the proposed method, and the difference is more for a small value of Δ in hypertext data. It has a very high ratio of MaxDuration/Δ, which signifies that the network is dense, and there is frequent communication among many of the nodes within a small span of time interval. This results in many more duration-wise maximal clique generation and their corresponding vertex expansion. The Himmel et al.'s method grows based on neighborhood creation at specific time intervals for each node, and as the number of nodes is less in the dense network, this boosts up the performance. Again, when the number of initial cliques is more (like in Infectious I), Virad et al. fails to proceed even if the dataset is sparse. The significant improvement in Δ-clique enumeration is seen in the performance for other datasets which are sparse. The scenario for sparse cases is more evident for the (Δ, γ)-clique enumeration case in Fig. 5. It can be seen if the Δ is increased for small γ, the improvement is more in the proposed method from the existing method of Banerjee & Pal. In the increased Δ, the initial duration-wise maximal clique set reduces the intermediate clique count, and the neighborhood in that large Δ also becomes smaller due to the sparse nature of the problem instance. Hence, we conclude that the proposed methodology significantly improves when the graph is sparse.

6 Conclusion and future directions

In this paper, we have proposed a methodology to enumerate all the maximal (Δ, γ)-cliques present in a temporal network. The proposed methodology has been analyzed for time and space requirements, and also, its correctness has been shown. To highlight its effectiveness, we have compared the execution time of the proposed methodology on five real-world publicly available datasets over the existing methods. As in many real-world applications, links are probabilistic in nature, so extending this study for such scenarios may be one possible future direction.

References

Akkoyunlu EA (1973) The enumeration of maximal cliques of large graphs. SIAM J Comput 2(1):1–6

Al-Naymat G (2008) Enumeration of maximal clique for mining spatial co-location patterns. In: 2008 IEEE/ACS International conference on computer systems and applications. IEEE, pp 126–133

Banerjee S, Pal B (2019) On the enumeration of maximal (Δ, γ)-cliques of a temporal network. In: Proceedings of the ACM India joint international conference on data science and management of data, COMAD/CODS 2019, Kolkata, India, January 3–5, 2019, pp 112–120

Banerjee S, Pal B (2021) A two-phase approach for enumeration of maximal (δ, γ)-cliques of a temporal network. In: International conference on database and expert systems applications. Springer, pp 346–357

Banerjee S, Pal B (2022) An efficient updation approach for enumerating maximal (δ, γ)-cliques of a temporal network. J Complex Netw 10(5):cnac027

Bentert M, Himmel AS, Molter H, Morik M, Niedermeier R, Saitenmacher R (2019) Listing all maximal k-Plexes in temporal graphs. J Exp Algorithmics (JEA) 24:1–27

Bhowmick SS, Seah BS (2015) Clustering and summarizing protein-protein interaction networks: a survey. IEEE Trans Knowl Data Eng 28(3):638–658

Bron C, Kerbosch J (1973) Algorithm 457: finding all cliques of an undirected graph. Commun ACM 16(9):575–577

Byun J, Woo S, Kim D (2019) Chronograph: enabling temporal graph traversals for efficient information diffusion analysis over time. IEEE Trans Knowl Data Eng 32(3):424–437

Chen Q, Fang C, Wang Z, Suo B, Li Z, Ives ZG (2016) Parallelizing maximal clique enumeration over graph data. In: International conference on database systems for advanced applications. Springer, pp 249–264

Chen Z, Yuan L, Lin X, Qin L, Yang J (2020) Efficient maximal balanced clique enumeration in signed networks. In: Proceedings of the web conference 2020:339–349

Cheng J, Ke Y, Fu AWC, Yu JX, Zhu L (2010) Finding maximal cliques in massive networks by h*-graph. In: Proceedings of the 2010 ACM SIGMOD international conference on management of data. ACM, pp 447–458

Cheng J, Ke Y, Fu AWC, Yu JX, Zhu L (2011) Finding maximal cliques in massive networks. ACM Trans Database Syst (TODS) 36(4):21

Cheng J, Zhu L, Ke Y, Chu S (2012) Fast algorithms for maximal clique enumeration with limited memory. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 1240–1248

Dai Q, Li RH, Liao M, Chen H, Wang G (2022) Fast maximal clique enumeration on uncertain graphs: A pivot-based approach. In: Proceedings of the 2022 international conference on management of data, pp 2034–2047

Dai Q, Li RH, Liao M, Wang G (2023) Maximal defective clique enumeration. Proc ACM Manag Data 1(1):1–26

Eppstein D, Löffler M, Strash D (2013) Listing all maximal cliques in large sparse real-world graphs. J Exp Algorithmics (JEA) 18:3

Ficara A, Cavallaro L, Curreri F, Fiumara G, De Meo P, Bagdasar O, Song W, Liotta A (2021) Criminal networks analysis in missing data scenarios through graph distances. PLoS ONE 16(8):e0255067

Garey MR, Johnson DS (2002) Computers and intractability, vol 29. W. H. Freeman, New York

Himmel AS, Molter H, Niedermeier R, Sorge M (2016) Enumerating maximal cliques in temporal graphs. In: Advances in social networks analysis and mining (ASONAM), 2016 IEEE/ACM international conference on. IEEE, pp 337–344

Himmel AS, Molter H, Niedermeier R, Sorge M (2017) Adapting the Bron–Kerbosch algorithm for enumerating maximal cliques in temporal graphs. Soc Netw Anal Min 7(1):35

Holme P, Saramäki J (2012) Temporal networks. Phys Rep 519(3):97–125

Hou B, Wang Z, Chen Q, Suo B, Fang C, Li Z, Ives ZG (2016) Efficient maximal clique enumeration over graph data. Data Sci Eng 1(4):219–230

Hulovatyy Y, Chen H, Milenković T (2015) Exploring the structure and function of temporal networks with dynamic graphlets. Bioinformatics 31(12):i171–i180

Isella L, Stehlé J, Barrat A, Cattuto C, Pinton JF, Van den Broeck W (2011) What's in a crowd? Analysis of face-to-face behavioral networks. J Theor Biol 271(1):166–180

Kumar S, Spezzano F, Subrahmanian V, Faloutsos C (2016) Edge weight prediction in weighted signed networks. In: Data mining (ICDM), 2016 IEEE 16th international conference on. IEEE, pp 221–230

Kumar S, Hooi B, Makhija D, Kumar M, Faloutsos C, Subrahmanian V (2018) Rev2: fraudulent user prediction in rating platforms. In: Proceedings of the eleventh ACM international conference on web search and data mining. ACM, pp 333–341

Manoussakis G (2019) A new decomposition technique for maximal clique enumeration for sparse graphs. Theor Comput Sci 770:25–33

Manoussakis G (2023) Efficient maximal cliques enumeration in weakly closed graphs. arXiv preprint arXiv:2303.02390

Masuda N, Holme P (2017) Temporal network epidemiology. Springer, Berlin

Mertzios GB, Molter H, Zamaraev V (2021) Sliding window temporal graph coloring. J Comput Syst Sci 120:97–115

Molter H, Niedermeier R, Renken M (2019) Enumerating isolated cliques in temporal networks. In: International conference on complex networks and their applications. Springer, pp 519–531

Mukherjee AP, Xu P, Tirthapura S (2016) Enumeration of maximal cliques from an uncertain graph. IEEE Trans Knowl Data Eng 29(3):543–555

Panzarasa P, Opsahl T, Carley KM (2009) Patterns and dynamics of users' behavior and interaction: network analysis of an online community. J Am Soc Inf Sci 60(5):911–932

Qin H, Li R, Yuan Y, Wang G, Yang W, Qin L (2020) Periodic communities mining in temporal networks: concepts and algorithms. IEEE Trans Knowl Data Eng 34:3927–3945

Rossi RA, Gleich DF, Gebremedhin AH, Patwary MMA (2014) Fast maximum clique algorithms for large graphs. In: Proceedings of the 23rd international conference on World Wide Web. ACM, pp 365–366

Rossi RA, Gleich DF, Gebremedhin AH (2015) Parallel maximum clique algorithms with applications to network analysis. SIAM J Sci Comput 37(5):C589–C616

Rozenshtein P, Gionis A (2019) Mining temporal networks. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data Mining, pp 3225–3226

Schmidt MC, Samatova NF, Thomas K, Park BH (2009) A scalable, parallel algorithm for maximal clique enumeration. J Parallel Distrib Comput 69(4):417–428

Viard J, Latapy M, Magnien C (2015) Revealing contact patterns among high-school students using maximal cliques in link streams. In: Proceedings of the 2015 IEEE/ACM international conference on advances in social networks analysis and mining 2015. ACM, pp 1517–1522

Viard T, Latapy M, Magnien C (2016) Computing maximal cliques in link streams. Theor Comput Sci 609:245–252

Viard T, Magnien C, Latapy M (2018) Enumerating maximal cliques in link streams with durations. Inf Process Lett 133:44–48

Xiang J, Guo C, Aboulnaga A (2013) Scalable maximum clique computation using mapreduce. In: 2013 IEEE 29th International conference on data engineering (ICDE). IEEE, pp 74–85

Zou Z, Li J, Gao H, Zhang S (2010) Finding top-k maximal cliques in an uncertain graph. In: 2010 IEEE 26th International conference on data engineering (ICDE 2010). IEEE, pp 649–652

Zschoche P (2022) A faster parameterized algorithm for temporal matching. Inf Process Lett 174:106181

Zschoche P, Fluschnik T, Molter H, Niedermeier R (2020) The complexity of finding small separators in temporal graphs. J Comput Syst Sci 107:72–92