Network Science **9** (S1): S83–S105, 2021
doi:10.1017/nws.2020.38

ORIGINAL ARTICLE

Isolation concepts applied to temporal clique enumeration*

Hendrik Molter*, Rolf Niedermeier and Malte Renken

TU Berlin, Faculty IV, Algorithmics and Computational Complexity (emails: rolf.niedermeier@tu-berlin.de, m.renken@tu-berlin.de)
*Corresponding author. Email: h.molter@tu-berlin.de

Special Issue Editor: Hocine Cherifi

Abstract
Isolation is a concept originally conceived in the context of clique enumeration in static networks, mostly used to model communities that do not have much contact to the outside world. Herein, a clique is considered *isolated* if it has few edges connecting it to the rest of the graph. Motivated by recent work on enumerating cliques in temporal networks, we transform the isolation concept to the temporal setting. We discover that the addition of the time dimension leads to six distinct natural isolation concepts. Our main contribution is the development of parameterized enumeration algorithms for five of these six isolation types for clique enumeration, employing the parameter "degree of isolation." In a nutshell, this means that the more isolated these cliques are, the faster we can find them. On the empirical side, we implemented and tested these algorithms on (temporal) social network data, obtaining encouraging results.

Keywords: community detection; dense subgraphs; social network analysis; time-evolving data; enumeration algorithms; fixed-parameter tractability

1. Introduction

"*Isolation is the one sure way to human happiness.*"
– Glenn Gould

Clique detection and enumeration is a fundamental primitive of complex network analysis. In particular, there are numerous approaches (both from a more theory-based and from a more heuristic side) for listing all maximal cliques (i.e. fully connected subgraphs) in a graph.[1] It is well known that finding a maximum-cardinality clique is computationally hard (NP-hard Karp, 1972), hard in the polynomial-time approximation sense (Håstad, 1999) and hard in the parameterized sense when parameterized by the clique cardinality (Downey & Fellows, 2013)). Hence, heuristic approaches usually govern computational approaches to clique finding and enumeration. We focus on the case of enumerating maximal cliques, that is, cliques that cannot be extended by adding further vertices. There have been numerous efforts to provide both theoretical guarantees and practically useful algorithms (Eppstein & Strash, 2013; Hüffner et al., 2009; Ito & Iwama, 2009; Komusiewicz et al., 2009; Tomita et al., 2006). In particular, to simplify (in a

*An extended abstract of this work appeared under the title "Enumerating Isolated Cliques in Temporal Networks" in the proceedings of the 8th International Conference on Complex Networks and their Applications (Molter et al., 2019). This version contains full proof details and a comprehensive empirical evaluation. This work was supported by the DFG, project MATE (NI 369/17).

computational sense) the task on the one hand and to enumerate more meaningful maximal cliques (for specific application contexts) on the other hand, Ito & Iwama (2009) introduced and investigated the enumeration of maximal cliques that are "isolated." Roughly speaking, isolation means that the connection of the maximal clique to the rest of the graph is limited, that is, there are few edges with one endpoint in the clique and one endpoint outside the clique; indeed, the degree of isolation can be controlled by choosing specific values of a corresponding isolation parameter. For instance, think of social networks where one wants to spot more or less segregated subcommunities with little interaction to the world outside but intensive interaction inside the community. We mention in passing that, recently, there have been (only) theoretical studies on the related concept of "secludedness" (van Bevern *et al.*, 2018; Chechik et al., 2017; Fomin et al., 2017; Golovach et al., 2020) which is somewhat similar to the isolation concept: whereas for isolation one requests "few outgoing edges," for secludedness one asks for "few outside neighbors"; while finding isolated cliques becomes tractable (Ito & Iwama, 2009), finding secluded ones remains computationally hard (van Bevern *et al.*, 2018).

Ito & Iwama (2009) showed that in static networks, isolated cliques often can be enumerated efficiently; the only exponential factor in the running time depends on the "isolation parameter," and so fairly isolated cliques can be enumerated quite quickly. In follow-up work, the isolation concept then was extended and more thorough experimental studies (also with financial networks) have been performed (Hüffner et al., 2009; Komusiewicz et al., 2009). However, analyzing complex networks more and more means studying time-evolving networks. Hence, computational problems known from static networks also need to be solved on temporal networks (mathematically, these are graphs with fixed vertex set but an edge set that changes over discrete time steps) (Holme & Saramäki, 2019; Latapy et al., 2018; Michail, 2016; Rossetti & Cazabet, 2018). Thus, not surprisingly, the enumeration of maximal cliques has recently been studied in the temporal setting (Bentert et al., 2019; Himmel et al., 2017; Viard et al., 2016, 2018). While getting algorithmically more challenging than in the static network case, the empirical results that have been achieved are encouraging. In this work, we now extend these studies by proposing to apply the isolation concept to the problem of temporal clique enumeration.

Since we believe that enumerating isolated cliques has its most important applications in community detection scenarios, we focus on only two of three basic isolation concepts described by Komusiewicz *et al.* (2009) for the static setting. More specifically, we only consider "maximal isolation" (every vertex has small outdegree) and "average isolation" (vertices have small outdegree on average) but do not study "minimal isolation" (at least one vertex has small outdegree). Nevertheless, we still face a richer modeling than in the static case since isolation can happen in two "dimensions": vertices and time; for both, we can consider maximum and average isolation. For the time dimension, this corresponds to considering the maximum or average outdegree over the time steps. Furthermore, we can switch the order in which vertices and time are considered. With these distinctions, we end up with eight natural ways to model isolation, where two "pairs" of isolation models turn out to be equivalent, finally leaving six different temporal isolation concepts for further study.

Our main contributions are as follows: first, as indicated above, we do conceptual work by identifying six mathematically formalized concepts of isolation for temporal networks. Second, building on and extending the algorithmic framework of Komusiewicz *et al.* (2009) for static networks, for small isolation values we provide efficient algorithms for five of our six isolated clique enumeration models and prove worst-case performance bounds for them (Theorem 7).[2] In this context, a main algorithmic contribution is the development of tailored subroutines (that are only partially shared between different isolation concepts). We leave the sixth case open for future research. Finally, on the empirical side, we contribute an encouraging experimental analysis (in the spirit of proof of concepts) of our algorithms based on real-world social network data. Our preliminary experiments indicate differences (mostly in terms of running time) but also (sometimes surprising) accordances between the concepts.

2. Preliminaries

In this section, we first provide some basic notation and terminology. We then recall the isolation concept for static graphs and transfer it to temporal graphs. Lastly, we give some motivating examples that are tailored to the arising different temporal isolation concepts and try to give an intuitive understanding of the differences between the various temporal isolation models.

Static graphs. Graphs in this paper are assumed to be undirected and simple. To clearly distinguish them from temporal graphs, they are sometimes referred to as *static* graphs. Let $G = (V, E)$ be a static graph. We denote the vertex set of G with $V(G)$ and the edge set of G with $E(G)$. For $v \in V(G)$, we use $N_G(v) := \{w \mid \{v, w\} \in E(G)\}$ and $N_G[v] := N_G(v) \cup \{v\}$ for the open and closed neighborhood and $\deg_G(v) := |N_G(v)|$ for the number of edges ending at v. For $v \in A \subseteq V$, $\operatorname{outdeg}_G(v, A) := |E \cap (\{v\} \times (V \setminus A))|$ denotes the number of edges connecting v to vertices in $V \setminus A$. Further, $\operatorname{outdeg}_G(A) := \sum_{v \in A} \operatorname{outdeg}_G(v, A)$. We use $\delta_G(A) := \min_{v \in A} \deg_G(v)$ for the minimum degree of the vertices in A. In all these notations, we omit the index G if there is no ambiguity.

Temporal graphs and temporal cliques. A *temporal graph* is a tuple $\mathcal{G} = (V, E_1, \ldots, E_\tau)$ of a vertex set V and τ edge sets $E_i \subseteq \binom{V}{2}$. The graphs $G_i := (V, E_i)$ are called the *layers* of $\mathcal{G}$. The *time edge set* $\mathcal{E}(\mathcal{G})$ (or $\mathcal{E}$ if $\mathcal{G}$ is clear from the context) is the disjoint union $\biguplus_{t=1}^{\tau} E_i$ of the edge sets of the layers of $\mathcal{G}$. For any $1 \leq a \leq b \leq \tau$, we define the (static) graphs $\bigcup_{t=a}^{b} G_t := \left(V, \bigcup_{t=a}^{b} E_t\right)$ and $\bigcap_{t=a}^{b} G_t := \left(V, \bigcap_{t=a}^{b} E_t\right)$.

Following the definition of Viard *et al.* (2016), a Δ*-clique* (for some $\Delta \in \mathbb{N}$) of $\mathcal{G}$ is a tuple $(C, [a, b])$ with $C \subseteq V$ and $1 \leq a \leq b \leq \tau$ such that C is a clique in $\bigcup_{i=t}^{t+\Delta} G_i$ for all $t \in [a, b - \Delta]$.

One easily observes that $(C, [a, b])$ is a Δ-clique in $\mathcal{G}$ if and only if $(C, [a, b - \Delta])$ is a 0-clique in $\mathcal{G}' = \left(V, E'_1, \ldots, E'_{\tau-\Delta}\right)$ where $E'_i := \bigcup_{t=i}^{i+\Delta} E_t$. Due to this, in our theoretical results, we will only concern ourselves with $\Delta = 0$ and simply refer to 0-cliques as *temporal cliques*.

Temporal isolation. We first introduce the isolation concepts for static graphs and then describe how we transfer them to the temporal setting. In a (static) graph G, a clique $C \subseteq V(G)$ is called *avg-c-isolated* if $\operatorname{outdeg}_G(C) < c \cdot |C|$ where $c \in \mathbb{Q}$ is some positive number (Ito & Iwama, 2009). Further, it is called *max-c-isolated* if $\max_{v \in C} \operatorname{outdeg}_G(v, C) < c$ Komusiewicz et al. (2009). Clearly, max-c-isolation implies avg-c-isolation.

Moving to temporal graphs, we aim at defining an isolation concept for temporal cliques. Recall that a temporal clique consists of a vertex set and a time interval. We apply the isolation requirement both on a vertex and on a time level, meaning that for each dimension, we can either require the average outdegree (as for static avg-c-isolation) or the maximum outdegree (as for static max-c-isolation) to be small. To make this more clear, we next provide some examples. For instance, we can require that, on average over all layers, the maximum outdegree in a layer is small. Or we can require that the average outdegree must be small in every single layer. Note that the ordering of the requirements for the time dimension and the vertex dimension also matters. Requiring the average outdegree to be small in every layer is different from requiring that, on average over all vertices, the maximum degree over all time steps must be small. Having two isolation requirements (avg and max) for two dimensions with two possible orderings, we arrive at eight canonical temporal isolation types. However, it turns out that if we use the same requirement for both dimensions, they behave commutatively, so it boils down to six *different* temporal isolation types. In the following, we give a formal definition for each of the six temporal isolation types. To make the names less confusing, we use "usually" to refer to the avg isolation requirement in the time dimension and "alltime" to refer to the max isolation requirement in the time dimension.

Definition 1 (Temporal isolation). *Let $c \in \mathbb{Q}$. A temporal clique $(C, [a, b])$ in a temporal graph $\mathcal{G} = (V, E_1, \ldots, E_\tau)$ is called*

- alltime-avg-c-isolated *if* $\max_{i\in[a,b]} \sum_{v\in C} \operatorname{outdeg}_{G_i}(v, C) < c \cdot |C|$,
- alltime-max-c-isolated *if* $\max_{v\in C} \max_{i\in[a,b]} \operatorname{outdeg}_{G_i}(v, C) < c$,
- avg-alltime-c-isolated *if* $\sum_{v\in C} \max_{i\in[a,b]} \operatorname{outdeg}_{G_i}(v, C) < c \cdot |C|$,
- max-usually-c-isolated *if* $\max_{v\in C} \sum_{i=a}^{b} \operatorname{outdeg}_{G_i}(v, C) < c \cdot (b + 1 - a)$,
- usually-avg-c-isolated *if* $\sum_{i=a}^{b} \sum_{v\in C} \operatorname{outdeg}_{G_i}(v, C) < c \cdot |C| \cdot (b + 1 - a)$, *and*
- usually-max-c-isolated *if* $\sum_{i=a}^{b} \max_{v\in C} \operatorname{outdeg}_{G_i}(v, C) < c \cdot (b + 1 - a)$.

We define the set of all six *isolation types* as $\mathcal{I} = \{$alltime-max, alltime-avg, max-usually, usually-avg, avg-alltime, usually-max$\}$.

For all isolation types $I \in \mathcal{I}$, an I-c-isolated temporal clique $(C, [a, b])$ is called *time-maximal* if there is no other I-c-isolated clique $(C', [a', b'])$ with $C' \supseteq C$ and $[a', b'] \supset [a, b]$. If there is no I-c-isolated temporal clique $(C', [a', b'])$ with $C' \supset C$ and $[a', b'] \supseteq [a, b]$, then we call $(C, [a, b])$ *vertex-maximal*. We call $(C, [a, b])$ *maximal* if it is time-maximal and vertex-maximal.

Subsequently, we give some intuition about the different isolation concepts. Note that for sufficiently small c, they all converge to disallowing *any* outgoing edges. We start with the most restrictive and perhaps also most straightforward isolation type, that is, *alltime-max-isolation*. Here, all vertices are required to have little or no outside contact at all times – think of a quarantined group. Slightly, less restrictive is the notion of *avg-alltime-isolation*. Here, it would be possible to have some distinguished "bridge" vertices inside the clique with relatively much outside contact, as long as most vertices never have many outgoing edges. If we reorder the terms, then we obtain *alltime-avg-isolation*. In contrast to the previous case, now the set of "bridge" vertices may be different at any point in time. A typical situation where this could occur is that there is a low bandwidth connection between the clique and the rest of the graph, only allowing a limited number of communications to occur at any given moment. The next isolation concept, *usually-max-isolation*, can be seen as allowing short bursts of activity, in which some or even all vertices have many outgoing edges, as long as the entire clique is isolated most of the time. Again, if we reorder the terms, then we get a less restrictive concept (*max-usually-isolation*). Here, the bursts of activity may happen at different times for different vertices. Finally, *usually-avg-isolation* is the least restrictive of these notions, only limiting the total number of outside contacts over all vertices and layers that are part of the temporal clique.

3. Basic facts

We now prove some important facts that we will make use of in the correctness proofs of our algorithms. Our first observation concerns the relation between different types of isolation. It is easily checked using Definition 1 since the maximum outdegree is at least the average outdegree which is in turn at least the outdegree of the intersection graph $\bigcap_{t=a}^{b} G_t$.

Observation 1. *Let $\mathcal{G} = (V, E_1, \ldots, E_\tau)$ be a temporal graph. The following nine implications hold for any $a \le b$, any clique C in $\bigcap_{t=a}^{b} G_t$, and any $c > 0$:*

$$\begin{tikzcd}
(C,[a,b])\ \textit{alltime-max-c-isolated} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & (C,[a,b])\ \textit{avg-alltime-c-isolated} \arrow[d, Rightarrow] \\
(C,[a,b])\ \textit{usually-max-c-isolated} \arrow[d, Rightarrow] & (C,[a,b])\ \textit{alltime-avg-c-isolated} \arrow[d, Rightarrow] \\
(C,[a,b])\ \textit{max-usually-c-isolated} \arrow[r, Rightarrow] \arrow[d, Rightarrow] & (C,[a,b])\ \textit{usually-avg-c-isolated} \arrow[d, Rightarrow] \\
C \textit{ is max-c-isolated in } \bigcap_{i=a}^{b} G_i \arrow[r, Rightarrow] & C \textit{ is avg-c-isolated in } \bigcap_{i=a}^{b} G_i
\end{tikzcd}$$

Note that Observation 1 does not hold for *maximal* isolated temporal cliques, for example, a maximal alltime-max-c-isolated clique (i.e. an alltime-max-c-isolated clique not contained in any bigger alltime-max-c-isolated clique) is not necessarily a maximal usually-avg-c-isolated clique.

Next, we state two lemmata limiting the minimal size of isolated cliques, helping us to confine the search space for our algorithms. They are inspired by the ideas employed by Komusiewicz *et al.* (2009) in the static setting.

Lemma 2. *Let G be a static graph and let C be a clique in G. Then, any avg-c-isolated subset $C' \subseteq C$ has size $|C'| > \delta(C) - c + 1$.*

Proof. Suppose $|C'| \le \delta(C) - c + 1$. Then any vertex $w \in C'$ has $\operatorname{outdeg}(w, C') = \deg(w) - (|C'| - 1) \ge \delta(C) - (|C'| - 1) \ge c$. Thus, C' is not avg-c-isolated. □

Lemma 3. *Let C be a clique in $G_\cap := \bigcap_{i=1}^{t} G_i$. Then any subset $C' \subseteq C$ for which $(C', [1, t])$ is usually-avg-c-isolated has size $|C'| > \delta_{G_\cap}(C) - c + 1$.*

Proof. Suppose not. Then $|C'| \le \delta_{G_\cap}(C) - c + 1 \le \delta_{G_i}(C) - c + 1$ for all i. By Lemma 2, C' is not avg-c-isolated in any G_i and thus certainly not usually-avg-c-isolated. □

Next, we show that some vertices must always be contained in vertex-maximal (and thus also in maximal) isolated cliques. This will allow us to refrain from searching for maximal isolated cliques that do not contain these vertices. We start with a technical lemma that will simplify the proof of the following result.

Lemma 4. *Let C be a clique in $G_\cap := \bigcap_{i=1}^{t} G_i$ and let $C' \subseteq C$ be such that $(C', [1, t])$ is a vertex-maximal usually-avg-c-isolated temporal clique. Then C' contains all vertices $v \in C$ that fulfill*

$$\sum_{i=1}^{t} \deg_{G_i}(v) \le t(\delta_{G_\cap}(C) + |C'| + 1). \tag{$*$}$$

Proof. We prove this statement by contraposition. Suppose ($*$) holds for some $v \in C \setminus C'$. Let $k := |C|$ and $k' := |C'|$. Then, we have

$$\sum_{i=1}^{t} \operatorname{outdeg}_{G_i}(v, C) = \sum_{i=1}^{t} \big(\deg_{G_i}(v) - (k-1)\big) \le t(\delta_{G_\cap}(C) - k + k' + 2).$$

Since v has $k-k'-1$ edges to $C \setminus (C' \cup \{v\})$ and k' edges to C' we obtain

$$\begin{aligned}\sum_{i=1}^{t} \operatorname{outdeg}_{G_i}(C' \cup \{v\}) &= \sum_{i=1}^{t} \left(\operatorname{outdeg}_{G_i}(C') + (k-k'-1) + \operatorname{outdeg}_{G_i}(v, C) - k'\right) \\ &< t\left(ck' + \delta_{G_\cap}(C) - k' + 1\right) \\ &= t\left(c(k'+1) + \delta_{G_\cap}(C) - c - k' + 1\right) \\ &< ct(k'+1),\end{aligned}$$

where the last inequality is due to Lemma 3. Thus, $C' \cup \{v\}$ is usually-avg-c-isolated. □

Using Lemma 4, we can now prove the following lemma, which is central for upper-bounding our algorithm's running time.

Lemma 5. *Let C be a clique in $G_\cap := \bigcap_{i=1}^{t} G_i$ and let $C' \subseteq C$ such that $(C', [1,t])$ is a vertex-maximal usually-avg-c-isolated temporal clique. Assume $\tilde{k} := \delta_{G_\cap}(C) - c + 2 \geq 0$ and let $\tilde{C} \subseteq C$ consist of the $\tilde{k}$ vertices v with the lowest values of $\sum_{i=1}^{t} \deg_{G_i}(v)$. Then $\tilde{C} \subseteq C'$.*

Proof. Let $k' := |C'|$. Suppose for contradiction that there exists $u \in \tilde{C} \setminus C'$. Then, for each $v \in C' \setminus \tilde{C}$, we have

$$\sum_{i=1}^{t} \operatorname{outdeg}_{G_i}(v, C') = \sum_{i=1}^{t} \left(\deg_{G_i}(v) - k' + 1\right) \geq \sum_{i=1}^{t} \left(\deg_{G_i}(u) - k' + 1\right) > t\left(\delta_{G_\cap}(C) + 2\right),$$

where the last inequality is due to Lemma 4. Thus, we get

$$\sum_{i=1}^{t} \operatorname{outdeg}_{G_i}(C') = \sum_{v \in C'} \sum_{i=1}^{t} \operatorname{outdeg}_{G_i}(v, C') > k't\left(\delta_{G_\cap}(C) + 2\right) \geq k'tc\,,$$

contradicting the isolation of C'. □

We explicitly formulate the following special case of Lemma 5, which will be used for alltime-avg- and avg-alltime-isolation.

Lemma 6. *Let C be a clique and let $C' \subseteq C$ be a maximal avg-c-isolated subset. Let $\tilde{C} \subseteq C$ be the $\delta(C) - c + 2$ vertices of minimal degrees. Then $\tilde{C} \subseteq C'$.*

Proof. This is a special case of Lemma 5 for $t = 1$. □

4. Enumerating maximal isolated temporal cliques

In this section, we present efficient algorithms to enumerate maximal isolated temporal cliques for five out of the six introduced temporal isolation concepts (all except usually-max).[3] These algorithms have *fixed-parameter tractable* (FPT) running times for the isolation parameter c, that is, for fixed c, the running time is a polynomial whose degree does not depend on c.[4] Formally, we show the following central result of this work; the proof will be given in Section 4.3.

Theorem 7. *Let a temporal graph $\mathcal{G}$ with τ layers, an isolation type $I \in \mathcal{I} \setminus \{$usually-max$\}$, and an isolation parameter $c \in \mathbb{Q}$ be given. Then all maximal I-c-isolated temporal cliques in $\mathcal{G}$ can be enumerated in FPT-time for the isolation parameter c. The specific running times depend on I and are given in Table 1.*

Table 1. Asymptotic upper bounds for the running time of our maximal isolated temporal clique enumeration algorithms for the different temporal isolation types

alltime-avg	alltime-max	avg-alltime	max-usually	usually-avg
$c^c\tau^2 \cdot \lvert V\rvert \cdot \lvert\mathscr{E}\rvert$	$2.89^c c\tau \cdot \lvert\mathscr{E}\rvert$	$5.78^c c\tau \cdot \lvert\mathscr{E}\rvert$	$2.89^c c\tau^3 \cdot \lvert\mathscr{E}\rvert$	$5.78^c c\tau^3 \cdot \lvert\mathscr{E}\rvert$

Algorithm 1: Enumerating maximal I-c-isolated cliques for $I \in \mathscr{I} \setminus \{\text{usually-max}\}$

```
Input: A temporal graph $\mathscr{G} = (V, E_1, \ldots, E_\tau)$, a $c \in \mathbb{Q}$, and an isolation type
       $I \in \mathscr{I} \setminus \{\text{usually-max}\}$.
Output: All maximal $I$-$c$-isolated cliques in $\mathscr{G}$.
1  result ← ∅
2  foreach $a = 1 \ldots \tau$ do
3      foreach $b = a \ldots \tau$ do
           /* Here we are looking for cliques with lifetime $[a, b]$.                */
4          $G_\cap \leftarrow \bigcap_{i=a}^{b} G_i$
5          Sort the vertices by ascending degree in $G_\cap$
6          foreach vertex $v$ do
               /* Vertex $v$ is the pivot vertex.                                     */
7              $C_v \leftarrow$ candidate set for pivot $v$ after trimming stage (in $G_\cap$), see Section 3 of Ito
               & Iwama (2009)
8              $k \leftarrow \lfloor \deg_{G_\cap}(v) - c + 2 \rfloor$ /* By Lemma 2, all isolated cliques are at
                   least this large.                                                   */
9              $\mathscr{C} \leftarrow$ set of all maximal cliques of size at least $k$ in $C_v \subseteq G_\cap$
10             foreach $C \in \mathscr{C}$ do
11                 subsets ← $I$-isolatedSubsets($C$, $[a, b]$, $\deg_{G_\cap}(v)$)
12                 result ← result ∪ $\{(C, [a, b]) \mid C \in \text{subsets}\}$
13             end
14         end
15     end
16 end
17 foreach $(C, [a, b]) \in$ result do
18     if $I$-isMaximal($C$, $[a, b]$) then
19         output $(C, [a, b])$
20     end
21 end
```

Our algorithms build upon the fact that every maximal I-c-isolated temporal clique $(C, [a, b])$ is contained in some vertex-maximal c-isolated clique C' of $G_\cap := \bigcap_{t=a}^{b} G_t$ (by Obervartion 1). Thus, we may utilize some known results from the static case (Ito & Iwama, 2009; Komusiewicz et al., 2009). Algorithm 1 constitutes the top-level algorithm. Here, we iterate over all possible time windows $[a, b]$ and apply the so-called trimming procedure developed by Ito & Iwama (2009) to $G_\cap$ to obtain, for each so-called *pivot vertex* v, a set $C_v \subseteq N[v]$ containing all avg-c-isolated cliques of $G_\cap$ that contain v. Subsequently, we enumerate all maximal cliques within C_v and test each of them for maximal I-c-isolated subsets. For this step, we employ Lemmas 2, 5, and 6 to quickly skip over irrelevant subsets. The details depend on the choice of I, as does the strategy for the last step, that is, removing non-maximal elements from the result set. Remember that here we have to pay attention to both, time- and vertex-maximality.

Function 2: alltime-avg-isolatedSubsets(C, $[a, b]$, δ)

```
d := ⌊|C| − δ + c − 2⌋ /* We may only remove the top d vertices by
    Lemma 6.                                                        */
𝒟′ ← {∅} /* Set of candidate sets to remove from C                  */
result ← ∅
while 𝒟′ ≠ ∅ do
    𝒟 ← 𝒟′
    𝒟′ ← ∅
    foreach D ∈ 𝒟 do
        C′ ← C \ D /* Potential isolated clique                     */
        if ∃i ∈ [a, b]: $\sum_{v \in C'} \deg_{G_i}(v) \geq |C'| \cdot (|C'| - 1 + c)$ then
            take i to be smallest possible
            if |C′| > δ − c + 2 then
                E ← the d vertices of C′ that have the highest degrees in $G_i$
                𝒟′ ← 𝒟′ ∪ {D ∪ {e} | e ∈ E \ D}
            end
        end
        else
            result ← result ∪ {C′}
        end
    end
end
return result
```

We proceed by describing the subroutines `isolatedSubsets()` (Line 11 of Algorithm 1) and `isMaximal()` (Line 18 of Algorithm 1). Then, we prove correctness of our algorithms and, finally, analyze their running times.

4.1 Enumerating isolated subsets

We now discuss the `isolatedSubsets()` subroutine of Algorithm 1 (Line 11). While the details depend on the isolation type, there are two main flavors. For alltime-max-isolation (Function 3) and max-usually-isolation (Function 5), it is possible to determine a single vertex that must be removed in order to obtain an isolated subset. By repeatedly doing so, one either reaches an isolated subset or the size threshold given by Lemma 2. In particular, each maximal clique contains at most one maximal isolated subset.

For usually-avg-isolation (Function 6), avg-alltime-isolation (Function 4), and alltime-avg-isolation (Function 2), multiple vertices are removal candidates. However, their number is upper-bounded by Lemmas 5 and 6, respectively. We therefore build a search tree, iteratively exploring removal sets of growing size. The case of alltime-avg-isolation (Function 2) is somewhat special, as here the set of removal candidates is different for each layer.

4.2 Checking for maximality

We now discuss the `isMaximal()` subroutine of Algorithm 1 (Line 18), which in turn uses an `isVertexMaximal()` subroutine. Note that, while each temporal clique (C, $[a, b]$) returned by `isolatedSubsets()` is vertex-maximal within its respective set C_v, it may be not vertex-maximal with regard to the entire graph. Moreover, we need to check for maximality with regard to cliques with a larger time window. The naive approach of pairwise comparing all elements of the result

Function 3: alltime-max-isolatedSubsets(C, $[a, b]$, δ)

```
1  $\forall v \in C : s_v := \max_{i\in[a,b]} \deg_{G_i}(v)$
2  $k := \lfloor \delta - c + 2 \rfloor$ /* All isolated cliques have size at least $k$ by
       Lemma 2.                                                                  */
3  while $|C| \geq k$ do
       /* Remove offending vertices until we either succeed or fail.            */
4      if $\exists v \in C : s_v \geq |C| - 1 + c$ then
5          $C \leftarrow C \setminus \{v\}$
6      end
7      else
8          return $\{C\}$
9      end
10 end
11 return $\emptyset$
```

Function 4: avg-alltime-isolatedSubsets(C, $[a, b]$, δ)

```
1  $\forall v \in C : s_v := \max_{i\in[a,b]} \deg_{G_i}(v)$
2  $d := \lfloor |C| - \delta + c - 2 \rfloor$ /* We may only remove the top $d$ vertices by
       Lemma 6.                                                                  */
3  $\{v_i \mid 1 \leq i \leq d\}$ := the $d$ vertices in $C$ with the highest values of $s_v$
4  $\mathscr{D}' \leftarrow \{\emptyset\}$ /* Set of candidate sets to remove from $C$              */
5  result $\leftarrow \emptyset$
6  while $\mathscr{D}' \neq \emptyset$ do
7      $\mathscr{D} \leftarrow \mathscr{D}'$
8      $\mathscr{D}' \leftarrow \emptyset$
9      foreach $D \in \mathscr{D}$ do
10         $C' \leftarrow C \setminus D$ /* Potential isolated clique                       */
11         if $\sum_{v\in C'} s_v \geq |C'| \cdot (|C'| - 1 + c)$ then
12             $j := \max\{0, i \mid v_i \in D\}$
13             $\mathscr{D}' \leftarrow \mathscr{D}' \cup \{D \cup \{v_i\} \mid j < i \leq d\}$
14         end
15         else
16             result $\leftarrow$ result $\cup \{C'\}$
17         end
18     end
19 end
20 return result
```

set is feasible but inefficient. Instead, for alltime-max-isolation, alltime-avg-isolation, and avg-alltime-isolation, it is sufficient to only check whether the time window can be extended in either direction (see Function 7), and whether a larger clique exists within the same time window (i.e. checking vertex-maximality). Except for the case of alltime-avg-isolation (Function 9), the latter can again be implemented more efficiently than using pairwise comparisons (Function 10). We modify the maximality test developed by Komusiewicz *et al.* (2009) which searches for cliques within the common neighborhood of C and then checks whether these can be used to build a larger isolated clique.

Function 5: max-usually-isolatedSubsets(C, $[a, b]$, δ)

```
1  $\forall v \in C : s_v := \sum_{i\in[a,b]} \deg_{G_i}(v)$
2  $k := \lfloor \delta - c + 2 \rfloor$ /* All isolated cliques have size at least $k$ by
       Lemma 2.                                                              */
3  while $|C| \geq k$ do
       /* Remove offending vertices until we either succeed or fail.         */
4      if $\exists v \in C : s_v \geq (b - a + 1)(|C| - 1 + c)$ then
5          $C \leftarrow C \setminus \{v\}$
6      end
7      else
8          return $\{C\}$
9      end
10 end
11 return $\emptyset$
```

Function 6: usually-avg-isolatedSubsets(C, $[a, b]$, δ)

```
1  $\forall v \in C : s_v := \sum_{i\in[a,b]} \deg_{G_i}(v)$
2  $d := \lfloor |C| - \delta + c - 2 \rfloor$ /* By Lemma 5, we can only remove the top $d$ vertices.
       */
3  $\{v_i \mid 1 \leq i \leq d\}$ := the $d$ vertices in $C$ with the highest values of $s_v$
4  $\mathscr{D}' \leftarrow \{\emptyset\}$ /* Set of candidate sets to remove from $C$         */
5  result $\leftarrow \emptyset$
6  while $\mathscr{D}' \neq \emptyset$ do
7      $\mathscr{D} \leftarrow \mathscr{D}'$
8      $\mathscr{D}' \leftarrow \emptyset$
9      foreach $D \in \mathscr{D}$ do
10         $C' \leftarrow C \setminus D$ /* Potential isolated clique                  */
11         if $\sum_{v\in C'} s_v \geq (b - a + 1) \cdot |C'| \cdot (|C'| - 1 + c)$ then
12             $j := \max\{0, k \mid v_k \in D\}$
13             $\mathscr{D}' \leftarrow \mathscr{D}' \cup \{D \cup \{v_k\} \mid j < k \leq d\}$
14         end
15         else
16             result $\leftarrow$ result $\cup \{C'\}$
17         end
18     end
19 end
20 return result
```

This modified vertex-maximality test also works for the cases of max-usually-isolation and usually-avg-isolation, but here we cannot avoid checking all time windows because isolation of $(C, [a, b])$ does not imply isolation of, say, $(C, [a, b - 1])$ (Function 8).

4.3 Correctness

We now show the correctness of our algorithms. We first prove that the `isolatedSubsets()` functions (Functions 2 to 6) behave as intended.

Function 7: I-isMaximal(C, $[a, b]$) - version for $I \in$ {alltime-avg, alltime-max, avg-alltime}

```
for $(a', b') \in \{(a-1, b), (a, b+1)\}$ do
    if $(C, [a', b'])$ is an isolated clique then
        return false
    end
end
return $I$-isVertexMaximal($C$, $[a, b]$)
```

Function 8: I-isMaximal(C, $[a, b]$) - version for $I \in$ {max-usually, usually-avg}

```
for $a' = a \ldots 1$ do
    if $C$ is not a clique in $G_{a'}$ then
        break
    end
    for $b' = b \ldots \tau$ do
        if $C$ is not a clique in $G_{b'}$ then
            break the inner loop
        end
        if $(C, [a', b'])$ $I$-$c$-isolated and $(a, b) \neq (a', b')$ then
            return false
        end
        if not $I$-isVertexMaximal($C$, $[a', b']$) then
            return false
        end
    end
end
return true
```

Function 9: alltime-avg-isVertexMaximal(C, $[a, b]$)

```
if the result set contains any $(C', [a, b])$ with $C' \supset C$ then
    return false
end
return true
```

Lemma 8. *Let $\mathcal{G} = (V, E_1, \ldots, E_\tau)$ be a temporal graph, $c \in \mathbb{Q}$, and $I \in \mathcal{I} \setminus$ {usually-max}. Let C be a clique in $G_\cap := \bigcap_{i=a}^{b} G_i$ and $\delta = \delta_{G_\cap}(C)$. Then I-*`isolatedSubsets`*(C, $[a, b]$, δ) returns all maximal sets $\tilde{C} \subseteq C$ such that $(\tilde{C}, [a, b])$ is I-c-isolated.*

Proof. For the sake of brevity, we will simply write that some set $X \subseteq C$ is, say, alltime-avg-isolated to denote that $(X, [a, b])$ is alltime-avg-isolated.

Case 1: $I =$ alltime-avg (Function 2). Let $\tilde{C} \subseteq C$ be any maximal subset which is alltime-avg-c-isolated, and suppose the function is currently checking C' with $\tilde{C} \subset C' \subseteq C$. Let $i \in [a, b]$ be the first layer in which C' is not avg-c-isolated. By Lemma 2 we have that $|C'| \geq |\tilde{C}| + 1 > \delta(C) - c + 2$, thus Lines 12 and 13 of Algorithm 2 are executed. Note that $\tilde{C}$ is avg-c-isolated in layer i, and let $\tilde{C}' \supseteq \tilde{C}$ be a maximal avg-c-isolated superset. Clearly, $\tilde{C} \subseteq \tilde{C}' \subset C'$. By Lemma 6, we have that $C' \setminus \tilde{C}'$ is a subset of the set E containing the d highest degree vertices of C' in layer i. Consequently, the function will add some set $C'' \subset C'$ with $C'' \supseteq \tilde{C}' \supseteq \tilde{C}$ to $\mathcal{D}'$. By recursively applying the same argument to C'', we deduce that the function will at some point find $\tilde{C}$.

Function 10: I-isVertexMaximal(C, $[a, b]$) - version for $I \in$ {alltime-max, avg-alltime, max-usually, usually-avg}

```
$G_\cap := \bigcap_{i=a}^{b} G_i$
$w := \arg\min_{v \in C} (\deg_{G_\cap}(v))$ /* w is the pivot of $(C, [a, b])$ */
$S := \{v \in N_{G_\cap}(w) \setminus C \mid N_{G_\cap}(v) \supseteq C\}$
$\mathscr{D}$ := set of all maximal cliques within $S \subset G_\cap$
for $D \in \mathscr{D}$ do
    while $(C \cup D, [a, b])$ not isolated do
        if $I \in \{\max - \text{usually}, \text{usually} - \text{avg}\}$ then
            $d \leftarrow \arg\max_{v \in D} ( \sum_{i=a}^{b} \deg_{G_i}(v))$
        end
        else if $I \in \{\text{alltime} - \max, \text{avg} - \text{alltime}\}$ then
            $d \leftarrow \arg\max_{v \in D} ( \max_{i=a}^{b} \deg_{G_i}(v))$
        end
        $D \leftarrow D \setminus \{d\}$
    end
    if $D \neq \emptyset$ then
        return false
    end
end
return true
```

Case 2: $I =$ **alltime-max (Function 3).** By Lemma 2 and Observation 1, we have that all alltime-max-c-isolated subsets of C have at least size k. If C contains an alltime-max-c-isolated subset $\tilde{C}$, then $\tilde{C}$ by definition does not contain any vertex v with $s_v \geq |\tilde{C}| - 1 + c$. Thus, by removing such vertices, we either reach the unique maximal alltime-max-c-isolated subset $\tilde{C}$ of C, or, if we reach size k, may conclude that no such subset exists.

Case 3: $I =$ **avg-alltime (Function 4).** Let $B := \{v_i \mid 1 \leq i \leq d\}$ and let $\tilde{C} \subseteq C$ be any maximal avg-alltime-c-isolated subset. Note that any subset of C is avg-alltime-c-isolated if and only if the same set was avg-c-isolated in a static graph where the degree of each vertex was $\max_{i \in [a,b]} \deg_{G_i}(v)$. By applying Lemma 6 to this auxiliary graph, we see that $\tilde{C}$ must contain $C \setminus B$. Thus, we observe analogously to Case 1 that the function will at some point reach $\tilde{C}$.

Case 4: $I =$ **max-usually (Function 5).** Works analogously to Case 2.

Case 5: $I =$ **usually-avg (Function 6).** Let $B := \{v_i \mid 1 \leq i \leq d\}$. By Lemma 5, we have that any maximal usually-avg-c-isolated subset of C must contain $C \setminus B$. Note that the loop generates all possible sets $D \subseteq B$ except those, for which $C \setminus D'$ has already found to be usually-avg-c-isolated for some $D' \subset D$. Therefore, all maximal usually-avg-c-isolated subsets of C are added to the result set. □

Next, we prove that the function `isVertexMaximal()` (Functions 9 and 10) behaves as intended.

Lemma 9. *Let $\mathscr{G} = (V, E_1, \ldots, E_\tau)$ be a temporal graph, $c \in \mathbb{Q}$, and $I \in \mathscr{I} \setminus \{$usually-max$\}$. Let $(C, [a, b])$ be an I-c-isolated clique in $\mathscr{G}$. Then I-`isVertexMaximal`$(C, [a, b])$ returns `true` if and only if $(C, [a, b])$ is a vertex-maximal I-c-isolated clique.*

Proof.

Case 1: $I =$ **alltime-avg (Function 9).** In this case, the function simply performs a pairwise comparison of all cliques in this time window and is thus trivially correct.

Case 2: $I \in \{$**alltime-max, avg-alltime, max-usually, usually-avg**$\}$ **(Function 10).** If the function returns `false`, then it has found a larger I-c-isolated clique $(C \cup D, [a, b])$. So suppose conversely that there is $C' \supset C$ for which $(C', [a, b])$ is an I-c-isolated clique. Then clearly, $C' \subseteq C \cup S$ and thus also $C' \subseteq C \cup D$ for some $D \in \mathscr{D}$. Let $x := |D \setminus C'| < |D|$ and let $X \subset D$ be the set of the first x vertices that the function removes from D. Then, it is not difficult to check for each of the four isolation types in question that $(C \cup D \setminus X, [a, b])$ is at least as I-isolated as $(C', [a, b])$. Thus, the function will not remove more than x vertices from D and instead return `false`. □

Lastly, we show that the function `isMaximal()` (Functions 7 and 8) behaves as intended.

Lemma 10. *Let $\mathscr{G} = (V, E_1, \ldots, E_\tau)$ be a temporal graph, let $c \in \mathbb{Q}$, and let $I \in \mathscr{I} \setminus \{$usually-max$\}$. Let $(C, [a, b])$ be an I-c-isolated clique in $\mathscr{G}$. Then, I-`isMaximal`$(C, [a, b])$ returns `true` if and only if $(C, [a, b])$ is a maximal I-c-isolated clique.*

Proof.
Case 1: $I \in \{$**alltime-max, alltime-avg, avg-alltime**$\}$ **(Function 7).** By Theorem 9, it only remains to show that the function returns `false` if there exists an I-c-isolated clique $(C', [a', b'])$ with $a' < a$ or $b' > b$. Suppose without loss of generality that $a' < a$. Then, $[a - 1, b] \subseteq [a', b']$ and thus $(C', [a - 1, b])$ is I-c-isolated.
Case 2: $I \in \{$**max-usually, usually-avg**$\}$ **(Function 8).** Since the function systematically tries all possible time windows $[a', b'] \subseteq [a, b]$ for which C is a clique, the correctness follows from Theorem 9. □

Now we have all the necessary pieces to prove the correctness of Algorithm 1.

Proposition 11 (Correctness of Algorithm 1). *Let $\mathscr{G} = (V, E_1, \ldots, E_\tau)$ be a temporal graph, let $c \in \mathbb{Q}$, and let $I \in \mathscr{I} \setminus \{$usually-max$\}$. Then, Algorithm 1 outputs exactly all maximal I-c-isolated temporal cliques.*

Proof. By Lemmas 8 and 10, every element of the output is in fact a maximal I-c-isolated clique. So it remains to show that all such cliques are in fact found by the algorithm. To this end, let $(C, [a, b])$ be any maximal I-c-isolated clique. Then, C is an avg-c-isolated clique in $G_\cap = \bigcap_{a \le i \le b} G_i$ by Observation 1. Ito & Iwama (2009) showed that we then have $C \subseteq C_v$ where $v \in C$ is of minimum degree. Further $|C| \ge |C_v| - k$ by Theorem 2. Thus, $C \subseteq C'$ for some $C' \in \mathscr{C}$, so $(C, [a, b])$ is added to the result set by Theorem 8. Finally, $(C, [a, b])$ is also included in the output by Theorem 10. □

4.4 Running time analysis

We will now estimate the time complexity of the different algorithms in terms of the numbers of layers τ, vertices $|V|$, and time edges $|\mathscr{E}|$, and the isolation parameter c.

We start estimating the running time of Algorithm 1 in terms of $T^{(I)}_{\texttt{isolatedSubsets()}}$ and $T^{(I)}_{\texttt{isMaximal()}}$, which shall denote the running times of the I-`isolatedSubsets()` and I-`isMaximal()` subroutines, respectively, since they are the parts of the running time that depend on the isolation type.

Lemma 12. *Let $\mathscr{G} = (V, E_1, \ldots, E_\tau)$ be a temporal graph, let $c \in \mathbb{Q}$, and let $I \in \mathscr{I} \setminus \{$usually-max$\}$. Algorithm 1 runs in $\mathscr{O}\big(2^c c^2 \tau \cdot |\mathscr{E}| + T^{(I)}_{\texttt{isolatedSubsets()}} + T^{(I)}_{\texttt{isMaximal()}}\big)$ time.*

Proof. We first investigate the running time of the first part of Algorithm 1 (the part up until Line 16). When iterating over all time windows $[a, b]$, each iteration of the inner loop extends the time window by one layer only. Because of this, we can compute the intersection graphs in $\mathcal{O}(\tau \cdot |\mathcal{E}|)$ time overall using incremental updates.

All sorting steps of the algorithm can be done by bucketsort using constant time per time window and pivot vertex. Computing C_v for all vertices of $G_\cap$ takes $\mathcal{O}(c^3 \cdot |E_\cap|)$ time Ito & Iwama (2009). Computing $\mathscr{C}$ from C_v takes $\mathcal{O}(|E(C_v)| + c \cdot |C_v| + 2^c c^2)$ time Komusiewicz et al. (2009). The overall number of steps (not counting `isolatedSubsets()` and `isMaximal()`) is thus in

$$\mathcal{O}\left(\tau \cdot |\mathcal{E}| + \sum_a \sum_b \left(c^3 \cdot |E_\cap| + \sum_{\text{pivot } v} (|E(C_v)| + c \cdot |C_v| + 2^c c^2)\right)\right).$$

Note that $\sum_a |E_\cap| \le \sum_a |E_a| \le |\mathcal{E}|$. Further, we assume that the algorithm is implemented to disregard vertices that have degree zero in G_a and thus also in $\bigcap_{t=a}^{b} G_t$. Because of this assumption, we can also record the following observation. If we sum $\deg_{G_a}(v) + 1$ over all time windows $[a, b]$ and pivot vertices v, then the result is at most

$$\sum_a \sum_b \sum_{\text{pivot } v} (\deg_{G_a}(v) + 1) \in \mathcal{O}\left(\sum_b \sum_a |E_a|\right) \subseteq \mathcal{O}(\tau \cdot |\mathcal{E}|)$$

by the handshake lemma. Of course, the same estimation is valid when summing over any of the following: $1 \le |C| \le |C_v| \le \deg_{G_\cap}(v) + 1 \le \deg_{G_a}(v) + 1$.

Another key observation is that $\sum_v |E(C_v)| \in \mathcal{O}(c^3 \cdot |E_\cap|)$ (Ito & Iwama, 2009). Using this, if we sum $|E(C_v)|$ over all time windows and pivot vertices, we obtain

$$\sum_a \sum_b \sum_{\text{pivot } v} |E(C_v)| \in \mathcal{O}\left(\tau \sum_a c^3 \cdot |E_\cap|\right) \subseteq \mathcal{O}\left(c^3 \tau \cdot |\mathcal{E}|\right).$$

Again, this also applies to $|E(C)| \le |E(C_v)|$.

Employing these observations, the above running time can be upper-bounded by

$$\mathcal{O}\left(\tau \cdot |\mathcal{E}| + c^3 \tau \cdot |\mathcal{E}| + c\tau \cdot |\mathcal{E}| + 2^c c^2 \tau \cdot |\mathcal{E}|\right) \subseteq \mathcal{O}\left(2^c c^2 \tau \cdot |\mathcal{E}|\right). \qquad \square$$

Now we analyze the running time $T^{(I)}_{\texttt{isolatedSubsets()}}$ of the I-`isolatedSubsets()` subroutine (Functions 2 to 6) depending on the isolation type I.

Lemma 13. $T^{(I)}_{\texttt{isolatedSubsets()}} \in \begin{cases} \mathcal{O}(2^c c\tau \cdot |\mathcal{E}|) & \text{if } I \in \{\textit{alltime-max}, \textit{max-usually}\}, \\ \mathcal{O}(2^c c^2 \tau \cdot |\mathcal{E}|) & \text{if } I \in \{\textit{usually-avg}, \textit{avg-alltime}\}, \\ \mathcal{O}(c^c \tau^2 \cdot |\mathcal{E}|) & \text{if } I \in \{\textit{alltime-avg}\}. \end{cases}$

Proof. Keep in mind the observations made in the proof of Lemma 12, which are also useful here. Additionally, note that within any iteration of Algorithm 1, $\mathscr{C}$ contains at most $2^{|C_v|-s}$ elements of size s for each $k \le s \le |C_v|$ and at most $2^{|C_v|-k} \le 2^c$ elements overall.

Case 1: $I \in \{$**alltime-max**, **max-usually**$\}$**.** Computing s_v takes $\mathcal{O}(\tau \cdot |\mathcal{E}|)$ overall (again, using incremental updating between time windows).

For each call, the loop runs at most $|C| - k \le |C| - (\delta + 1) + c \le c$ times, each needing $\mathcal{O}(|C|)$ time. Since there are $|\mathscr{C}|$ calls per time window and pivot, the overall time is in $\mathcal{O}(\tau \cdot |\mathcal{E}| + 2^c c\tau \cdot |\mathcal{E}|) \subseteq \mathcal{O}(2^c c\tau \cdot |\mathcal{E}|)$.

Case 2: $I =$ **alltime-avg.** There are d^d possible options for D, each tested in $\mathcal{O}(\tau)$ time. Of each size s, there are $2^{C_v - s}$ elements of that size within $\mathscr{C}$. Note that $d \le |C| + c - |C_v| - 1 \le c - 1$. Thus, the loop needs

$$\mathcal{O}\left(\sum_{s} 2^{C_v - s} d^d \tau\right) \subseteq \mathcal{O}\left(\sum_{s} 2^{C_v - s} c^{s+c-C_v-1}\right) \subseteq \mathcal{O}\left(\sum_{s} c^{c-1}\tau\right) \subseteq \mathcal{O}\left(c^c\tau\right)$$

time per time window and pivot, giving $\mathcal{O}(c^c\tau^2 \cdot |\mathcal{E}|)$ time overall.

Case 3: $I \in$ {**usually-avg**, **avg-alltime**}. Computing s_v again takes $\mathcal{O}(\tau \cdot |\mathcal{E}|)$ time overall. Here, there are 2^d possible options for D, each tested in constant time. Thus, the loop needs

$$\mathcal{O}\left(\sum_{s} 2^{C_v - s} 2^d\right) \subseteq \mathcal{O}\left(\sum_{s} 2^{C_v - s} 2^{s+c-C_v-1}\right) \subseteq \mathcal{O}\left(\sum_{s} 2^{c-1}\right) \subseteq \mathcal{O}\left(2^c c\right)$$

time per time window and pivot (again $s = |C|$). In total, this gives $\mathcal{O}(2^c c^2 \tau \cdot |\mathcal{E}|)$ time. □

Finally, we analyze the running time $T^{(I)}_{\texttt{isMaximal()}}$ of the `isMaximal()` subroutine (Functions 7 and 8) depending on the isolation type.

Lemma 14. $$T^{(I)}_{\texttt{isMaximal()}} \in \begin{cases} \mathcal{O}(2.89^c c\tau \cdot |\mathcal{E}|) & \textit{if } I = \textit{alltime-max}, \\ \mathcal{O}(2.89^c c\tau^3 \cdot |\mathcal{E}|) & \textit{if } I = \textit{max-usually}, \\ \mathcal{O}(2^{2c}\tau \cdot |V| \cdot |E|) & \textit{if } I = \textit{alltime-avg}, \\ \mathcal{O}(5.78^c c\tau \cdot |E|) & \textit{if } I = \textit{avg-alltime}, \\ \mathcal{O}(5.78^c c\tau^3 \cdot |E|) & \textit{if } I = \textit{usually-avg}. \end{cases}$$

Proof.
Case 1: $I =$ **alltime-max.** Each call to `isolatedSubsets()` returns at most one clique, thus for each time window and pivot v there are at most $|\mathcal{C}| \leq 2^c$ cliques to be checked. Each call to `isMaximal()` takes $\mathcal{O}(|C|)$ time, in addition to one call to `isVertexMaximal()`.

Regarding `isVertexMaximal()`, the size of $S \subseteq N(v) \setminus C$ is at most $\deg(v) + 1 - |C| < c$ and finding it takes $c \cdot |C|$ time. Computing the set of cliques $\mathcal{D}$ takes $\mathcal{O}(3^{c/3})$ time (Tomita et al., 2006) and $\mathcal{D}$ has size at most $3^{c/3}$. For each $D \in \mathcal{D}$, we need $\mathcal{O}(|D|) \subseteq \mathcal{O}(c)$ time.

Altogether, each call to isMaximal takes $\mathcal{O}(c \cdot |C| + 3^{c/3}c)$ time, giving an overall running time of

$$\mathcal{O}\left(2^c c\tau \cdot |\mathcal{E}| + 2^c 3^{c/3} c\tau \cdot |\mathcal{E}|\right) \subseteq \mathcal{O}\left(2.89^c c\tau \cdot |\mathcal{E}|\right).$$

Case 2: $I =$ **max-usually.** Again at most 2^c cliques need to be checked for each time window and pivot v.

For each call, we need $\mathcal{O}(\tau \cdot |E(C)|)$ to determine the layers where C is a clique and $\mathcal{O}(\tau^2 \cdot |C|)$ for the isolation check. Further, there are τ^2 calls to `isVertexMaximal()`. Of these, each takes $\mathcal{O}(3^{c/3}c)$ time as for alltime-max-isolation.

Thus, the total time per call is $\mathcal{O}(\tau|E(C)| + \tau^2|C| + 3^{c/3}c\tau^2)$, giving an overall time bound of

$$\mathcal{O}\left(2^c c^3 \tau^2 \cdot |\mathcal{E}| + 2^c \tau^3 \cdot |\mathcal{E}| + 2^c 3^{c/3} c\tau^3 \cdot |\mathcal{E}|\right) \subseteq \mathcal{O}\left(2.89^c c\tau^3 \cdot |\mathcal{E}|\right).$$

Case 3: $I =$ **alltime-avg.** Each call to `isolatedSubsets()` returns at most $2^d \leq 2^c$ cliques, therefore there are at most $2^c \cdot |\mathcal{C}| \leq 2^{2c}$ cliques to be checked per time window and pivot. Each call to `isMaximal()` takes $\mathcal{O}(|C|)$ time, in addition to one call to `isVertexMaximal()`.

Within `isVertexMaximal()`, we only need to check against cliques for the same time window, of these there are at most $\sum_v |\mathcal{C}|$ many, each of size at most $|C_v|$. The time needed to check a clique for maximality is linear in the total size of this set, that is, $\mathcal{O}(\sum_v |\mathcal{C}| \cdot |C_v|) \subseteq \mathcal{O}(|\mathcal{C}| \cdot |E_\cap|)$.

In total, each call to `isMaximal()` takes $\mathcal{O}(|\mathcal{C}| \cdot |E_\cap|)$ time, and the total time taken is thus $\mathcal{O}(2^{2c}\tau \cdot |V| \cdot |\mathcal{E}|)$.

Case 4: $I =$ **usually-avg.** Each call to `isolatedSubsets()` returns at most $2^d \leq 2^c$ cliques. Apart from this extra factor, the analysis is identical to the max-usually-isolation case. Thus, the total time is $\mathcal{O}(2.89^c 2^c c\tau^3 \cdot |\mathcal{E}|) \subseteq \mathcal{O}(5.78^c c\tau^3 \cdot |\mathcal{E}|)$.

Case 5: $I=$ **avg-alltime.** Each call to `isolatedSubsets()` returns at most $2^d \leq 2^c$ cliques. Apart from this extra factor, the analysis is identical to the alltime-max-isolation case. Thus, the total time is $\mathcal{O}(2.89^c 2^c c\tau \cdot |\mathcal{E}|) \subseteq \mathcal{O}(5.78^c c\tau \cdot |\mathcal{E}|)$. □

Now, it is straightforward to check that Lemmas 13 and 14 together with Lemma 12 imply the running times given in Table 1. This together with Proposition 11 completes the proof of Theorem 7.

We remark that we can derive a running time lower bound from a lower bound on the number of avg-c-isolated cliques in a graph by Ito & Iwama (2009). When investigating their proof, one can see that their lower bound actually holds also for the number of *max*-c-isolated *maximal* cliques in a graph. Note that this is also a lower bound on the number of maximal avg-c-isolated cliques and the number of maximal max-c-isolated cliques in a graph. Ito & Iwama (2009) showed that for every $n \in \mathbb{N}$ and $c \in \omega(\log n)$, there is a graph with n vertices for which the number of avg-c-isolated maximal cliques is superpolynomial in n. From this, we get the following running time lower bound for all considered temporally isolated clique enumeration variants where $\log^*$ is the extremely slow-growing iterated logarithm.

Proposition 15. *For all $I \in \mathcal{I}$, there exists no algorithm that enumerates all maximal I-c-isolated cliques in a given temporal graph $\mathcal{G}$ in $2^{\mathcal{O}(\frac{c}{\log^* c})} \cdot |\mathcal{G}|^{\mathcal{O}(1)}$ time for given $c \in \mathbb{Q}$.*

Proof. Ito & Iwama (2009) (Theorem 2.3) showed that for every $n \in \mathbb{N}$ and $c \in \omega(\log n)$, there is a graph $G_{n,c} = (V, E)$ on $|V| = n$ vertices for which the number of max-c-isolated maximal cliques is superpolynomial in n. Let $c = \log n \cdot \log^* n \in \omega(\log n)$ for some $n \in \mathbb{N}$ and let the graph $G_{n,c}$ be the one and only layer of temporal graph $\mathcal{G}$. Then the number of maximal I-c-isolated cliques in $\mathcal{G}$ is superpolynomial in n for all $I \in \mathcal{I}$.

Assume now for contradiction that there is an algorithm enumerating all maximal I-c-isolated cliques in $\mathcal{G}$ in $2^{\mathcal{O}(c/\log^* c)} \cdot |\mathcal{G}|^{\mathcal{O}(1)}$ time for some $I \in \mathcal{I}$. Note that this implies that there are at most $2^{ac/\log^* c} \cdot |\mathcal{G}|^b$ many I-c-isolated cliques in $\mathcal{G}$ for some constants $a, b \in \mathbb{N}$. If we now plug in our choice of c we get

$$2^{a\frac{c}{\log^* c}} \cdot |\mathcal{G}|^b = 2^{a\frac{\log n \cdot \log^* n}{\log^*(\log n \cdot \log^* n)}} \cdot |\mathcal{G}|^b \leq 2^{a\frac{\log n \cdot \log^* n}{\log^*(\log n)}} \cdot |\mathcal{G}|^b = 2^{a \log n \frac{\log^* n}{(\log^* n)-1}} \cdot |\mathcal{G}|^b.$$

For sufficiently large n, the fraction $\frac{\log^* n}{(\log^* n)-1}$ approaches 1. This implies that the number of maximal I-c-isolated cliques in $\mathcal{G}$ is polynomial in n, contradicting the above. □

Proposition 15 implies that our running times for $I \in \mathcal{I} \setminus \{$alltime-max, usually-max$\}$ cannot be improved significantly (see Table 1).

5. Experimental evaluation

In this section, we empirically evaluate the running times of our enumeration algorithms for maximal isolated Δ-cliques (Algorithm 1) on several real-world temporal networks. In particular, we investigate the effect of different isolation concepts as well as different values for isolation parameter c and Δ (see the definition of Δ-cliques in Section 2) on the running time and on the number of cliques that are enumerated. We also draw some comparisons concerning running times to a state-of-the-art algorithm to enumerate maximal (non-isolated) Δ-cliques by Bentert *et al.* (2019).

5.1 Setup and statistics

We implemented our algorithms[5] in Python 3.6.8 and carried out experiments on an Intel Xeon E5-1620 computer clocked at 3.6 GHz and with 64 GB RAM running Debian GNU/Linux 6.0.

Table 2. Statistics for the data sets used in our experiments. The lifetime L of a graph is the difference between the largest and smallest time stamp on an edge in the graph. The resolution r is the time between subsequent layers

Data Set	# Vertices $\|V\|$	# Edges $\|\mathscr{E}\|$	Resolution r (in s)	Lifetime L (in s)
highschool-2011	126	28,560	20	272,330
highschool-2012	180	45,047	20	729,500
highschool-2013	327	188,508	20	363,560
tij_pres_LH10	73	150,126	20	259,180

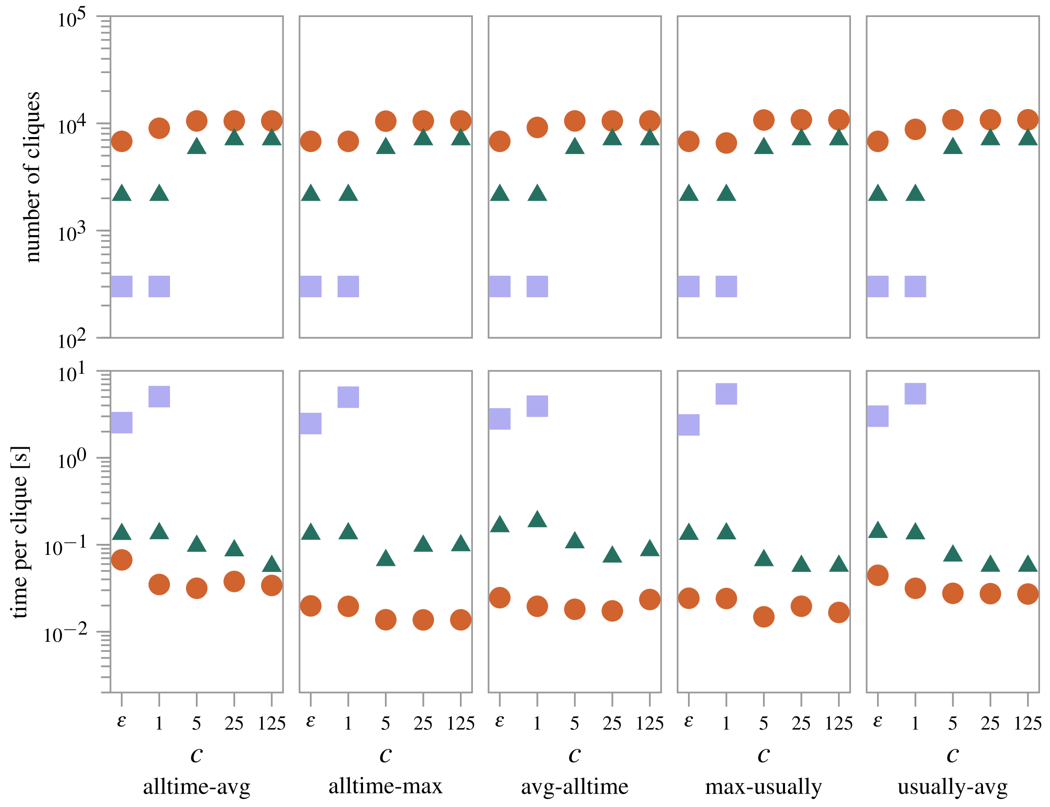

Figure 1. Plot for the data set "highschool-2011" showing the number of cliques (top) and the computation time per clique (bottom) for the different temporal isolation types and different values of c and Δ. The different Δ-values are visualized by the different markers, with circles, triangles and squares denoting Δ-values of 0, 5^3, and 5^5, respectively.

The given times refer to single-threaded computation. Bentert *et al.* (2019) implemented their algorithm in Python 2.7.12.

For the sake of comparability we tested our implementation on four freely available data sets, the first three of which were also used by Bentert *et al.* (2019):

- Face-to-face contacts between high school students ("highschool-2011", "highschool-2012", "highschool-2013" (Gemmetto et al., 2014; Stehlé et al., 2011; Fournet & Barrat, 2014)),
- Spatial proximity between persons in a hospital ("tij_pres_LH10" (Génois & Barrat, 2018)).

We list the most important statistics of the data set in Table 2. We chose five roughly exponentially increasing values $\varepsilon, 1, 5, 25, 125$ for the isolation parameter c, where $\varepsilon := 0.001$ effectively

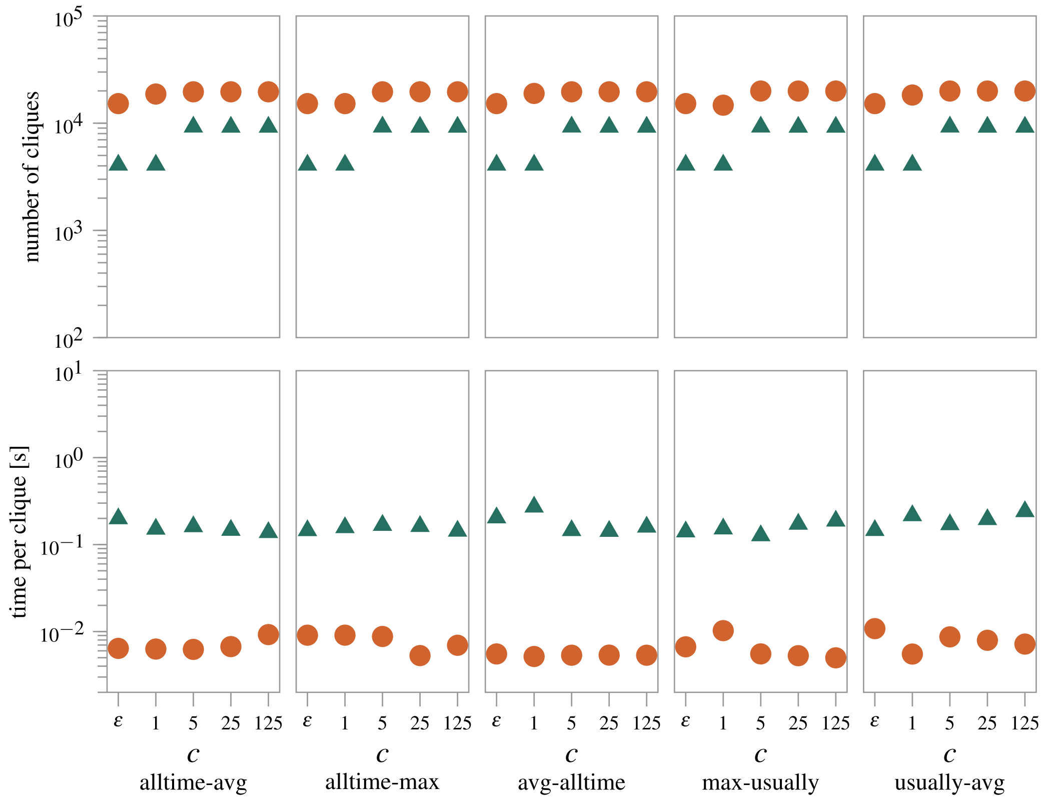

Figure 2. Plot for the data set "highschool-2012" (see also description of Figure 1).

requires complete isolation and $125 \approx |V|$ imposes little or no restriction. We chose our Δ-values in the same fashion as Bentert *et al.* (2019). In order to limit the influence of time scales in the data and to make running times comparable between instances, the chosen Δ-values of 0, 5^3, and 5^5 were scaled by $L/(5 \cdot |\mathscr{E}|)$, where L is the temporal graph's lifetime in seconds Himmel et al. (2017).

5.2 Experimental results

In Figures 1, 2, 3 and 4 the number of maximal isolated Δ-cliques and the running time are plotted for each of the five isolation types and a range of isolation values c. Missing values indicate that the respective instance exceeded the time limit of 1 hour. In general, the different isolation types produce surprisingly similar outputs. This suggests that the degrees of the vertices forming an isolated Δ-clique are typically rather similar and remain constant over the lifetime of the clique. Unsurprisingly, raising the value of c increases the number of maximal cliques as the isolation restriction is weakened. However, this effect ceases roughly at $c = 5$. Increasing c further does not produce additional cliques, suggesting that the vertices in Δ-cliques we found in the data sets mostly have out-degree at most five. Furthermore, we can generally observe that the number of maximal cliques decreases with increasing values of Δ, which might seem unexpected at first glance, but is a consequence from finding many small cliques (with few vertices as well as short time intervals) for small Δ-values that "merge together" for larger Δ-values. This behavior is consistent across all data sets we investigated.

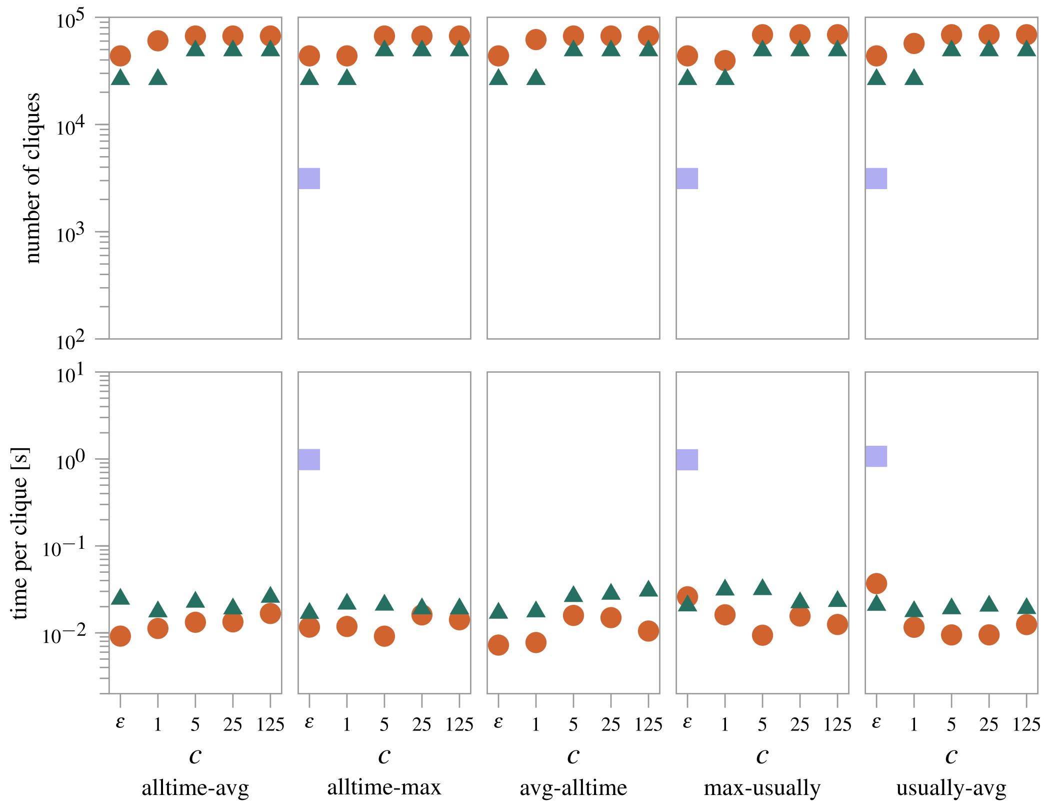

Figure 3. Plot for the data set "highschool-2013" (see also description of Figure 1).

Regarding running time, our algorithm is generally slower than the non-isolated clique enumeration algorithm by Bentert *et al.* (2019), even for small values of c. For comparison, the algorithm by Bentert *et al.* (2019) solved the instances "highschool-2011", "highschool-2012", and "highschool-2013" for the same values for Δ that we considered in less than 17 seconds per instance. We believe that the two main reasons for our algorithm to be slower are the following. On the one hand, the maximality check we perform is much more complicated than the one of the algorithm of Bentert *et al.* (2019), which is an issue that also occurs in the static case (Komusiewicz et al., 2009; Hüffner et al., 2009). On the other hand, we have to explicitly interate through more or less all possible intervals in which we could find an isolated Δ-clique, which seems unavoidable in our setting. A particular consequence of this is that our algorithm is not *output-sensitive*, that is, the running time can be much larger than the number of maximal isolated Δ-cliques in the input graph. In the case of (non-isolated) Δ-clique enumeration, there are ways to circumvent these issues and in particular, the algorithm of Bentert *et al.* (2019) is output-sensitive. Our algorithm and the algorithm of Bentert *et al.* (2019) have a similar running time behavior with respect to Δ, that is, the running time increases with Δ, once Δ reaches moderately large values. Since higher values of Δ create a more dense graph after the preprocessing step, this behavior is expected. The algorithm of Bentert *et al.* (2019) is slow for near-zero values of Δ (compared to itself for larger values of Δ). We do not observe this phenomenon in most of our algorithms. In the variants for max-usually and usually-avg, however, we experience a similar issue for small

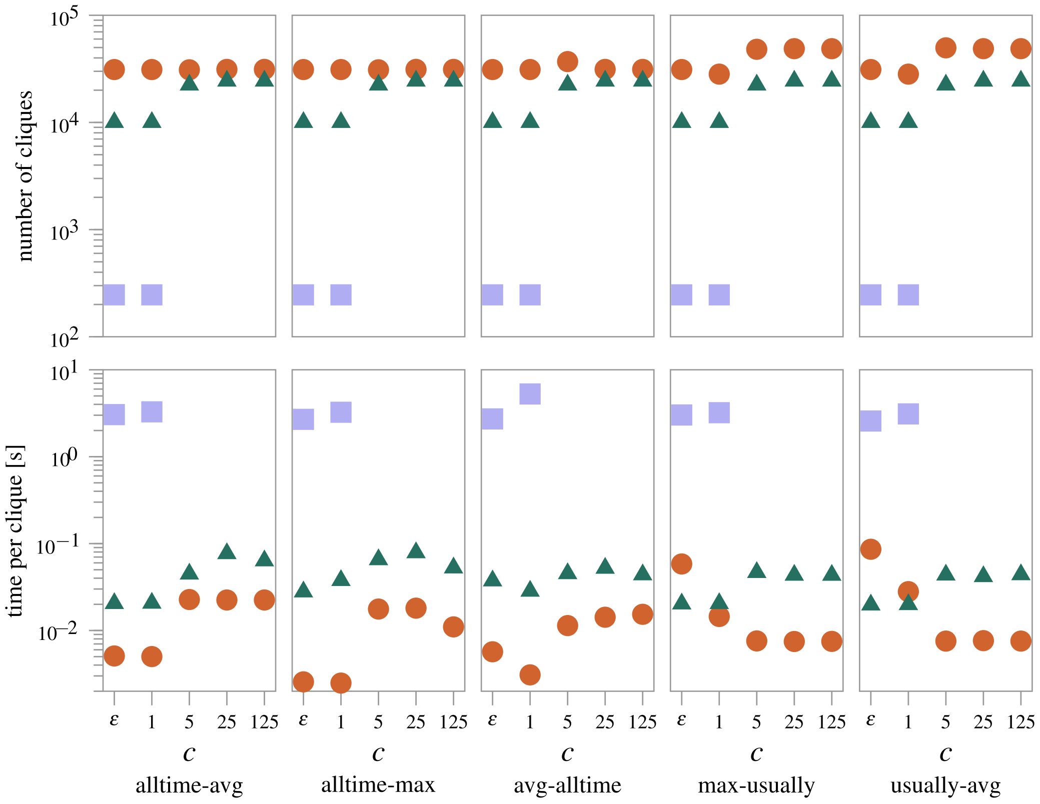

Figure 4. Plot for the data set "tij_pres_LH10" (see also description of Figure 1).

values of c, especially visible in the "tij_pres_LH10" data set (Figure 4), where the running time is surprisingly high for $\Delta = 0$ and $c = \varepsilon$. A possible explanation is that the "usually-variants" use a different maximality check than the "alltime-variants". Interestingly, no universal trend can be observed for the running time taken per resulting clique with respect to c, which stands in contrast to our theoretical worst-case running time analysis.

In terms of the number of cliques found, the number of maximal isolated cliques is generally lower than the number of maximal cliques, which is unsurprising but not obvious. For the Δ-values of 0 and 5^3, the number of maximal isolated cliques is about $1/3$ of the number of maximal cliques, whereas the results for $\Delta = 5^5$ suggest that the ratio is roughly $1/10$. Especially for these larger Δ-values it seems plausible that the isolated cliques are more significant than non-isolated cliques as they are much less likely to be the result of noise or data artifacts.

To get a more fine-grained picture with regard to Δ, we tested intermediate values for Δ and c representatively on the "tij_pres_LH10" data for avg-alltime-isolation. The results are shown in Figures 5 and 6. For increasing values of Δ, the number of cliques drops while the running time per clique rises. For fixed Δ and increasing c, the situation is very different. Here, both number of cliques and running time per clique quickly rise and subsequently level off around $c = 5$.

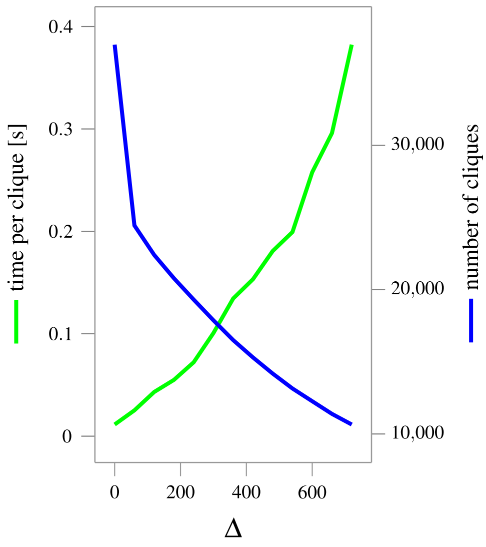

Figure 5. Number of cliques and running time for avg-alltime-isolation on the data set “tij_pres_LH10” with $\Delta = 5^3$.

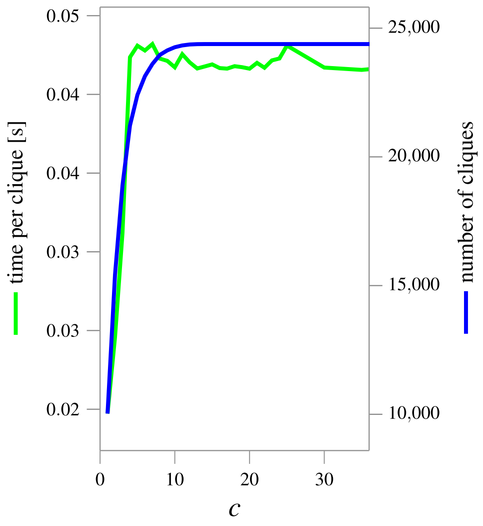

Figure 6. Number of cliques and running time for avg-alltime-isolation on the data set “tij_pres_LH10” with $c = 5$.

6. Conclusion

We have brought the concept of isolation from the static to the temporal graph setting, introducing six different types of temporal isolation. For five out of those we developed algorithms and showed that enumerating maximal temporally isolated cliques is fixed-parameter tractable with respect to the isolation parameter. This leaves one case (usually-max-isolation) open for future research on computational complexity classification. In this case, the main difficulty in adapting our algorithm is that we found no way to limit the running time required by the `isolatedSubsets()` subroutine significantly below $\Omega(2^{|V|})$.

As a rule of thumb, if there are no specific requests from the use case, we recommend to choose the alltime-max concept as a default, since overall it allowed for the fastest running times without huge differences in terms of enumerated maximal cliques.

From an algorithm engineering perspective there is still room for improvement. So far the running times make it hard to analyze larger data sets as done for example by Bentert *et al.* (2019) in the “non-isolated” setting. Another possibility to approach this issue it to shift focus from the enumeration of all maximal temporally isolated cliques to the “detection” problem, that is, to “only” search for one large temporally isolated clique (if one exists). Depending on the application, this might still be a task worth investigating. It could allow for better heuristic improvement such as pruning rules that remove parts of the input in which large cliques can be ruled out and would make the maximality check unnecessary.

Finally, as in the static case, it would be natural to apply the isolation concepts to further community models (dense subgraphs) such as for example temporal k-plexes[6] (Bentert et al., 2019; Komusiewicz et al., 2009). However, a major obstacle in this task is that the last row of implications from Observation 1 does not transfer to k-plexes and thus we cannot use the intersection graph of a temporal graph to search for candidate sets. This is due to the fact that the intersection of two k-plexes in not necessarily a k-plex. Hence, a fundamentally different approach is probably necessary to efficiently enumerate isolated temporal k-plexes.

Acknowledgments. We thank our student assistant Lilian Jacobs for the work on the implementation of our algorithms and the anonymous reviewers of the conference version for their helpful feedback. We further thank two anonymous reviewers of *Network Science* whose constructive feedback helped to significantly improve the presentation.

Conflicts of Interest. None.

Notes

1 *Network* and *graph* are used interchangeably.

2 In terms of the language of parameterized algorithmics (Cygan et al., 2015; Downey & Fellows, 2013; Flum & Grohe, 2006; Niedermeier, 2006), we show that these cases are fixed-parameter tractable when parameterized by isolation value.

3 Usually-max-isolation was dropped here since, even though the same approach as for the other isolation concepts also works for usually-max-isolation, we found no way to limit the work that would be required in the `isolatedSubsets()` subroutine significantly below $\Omega(2^{|V|})$.

4 The isolation parameter c only influences the constant factor of the polynomial running time but not the degree of the polynomial, that is, the running time is $f(c) \cdot \text{poly}(|\mathcal{G}|)$ for some computable function f.

5 The code of our implementation is freely available at https://www.akt.tu-berlin.de/menue/software/

6 A *k-plex* is a vertex set C such that each vertex in C is connected to all but at most k other vertices in C. In particular, a 1-plex is a clique. A *temporal k-plex* is a tuple $(C, [a, b])$ such that C is a k-plex in every layer G_i with $i \in [a, b]$.

References

Bentert, M., Himmel, A.-S., Molter, H., Morik, M., Niedermeier, R., & Saitenmacher, R. (2019). Listing All Maximal k-Plexes in Temporal Graphs. *ACM Journal of Experimental Algorithmics*, *24*(1), 1.13:1–1.13:27.

van Bevern, R., Fluschnik, T., Mertzios, G.B., Molter, H., Sorge, M., & Suchý, O. (2018). The parameterized complexity of finding secluded solutions to some classical optimization problems on graphs. *Discrete Optimization*, *30*, 20–50.

Chechik, S., Johnson, M.P., Parter, M., & Peleg, D. (2017). Secluded connectivity problems. *Algorithmica*, *79*(3), 708–741.

Cygan, M., Fomin, F.V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M. & Saurabh, S. (2015). *Parameterized algorithms*. Springer.

Downey, R.G., & Fellows, M.R. (2013). *Fundamentals of parameterized complexity*. Springer.

Eppstein, D., & Strash, D. (2013). Listing all maximal cliques in large sparse real-world graphs. *ACM Journal of Experimental Algorithmics*, *18*, 3.1:1–3.1:21.

Flum, J., & Grohe, M. (2006). *Parameterized complexity theory*. Texts in Theoretical Computer Science. An EATCS Series, vol. XIV. Springer.

Fomin, F.V., Golovach, P.A., Karpov, N., & Kulikov, A.S. (2017). Parameterized complexity of secluded connectivity problems. *Theory of Computing Systems*, *61*(3), 795–819.

Fournet, J., & Barrat, A. (2014). Contact patterns among high school students. *PLOS ONE*, *9*(9), 1–17.

Gemmetto, V., Barrat, A., & Cattuto, C. (2014). Mitigation of infectious disease at school: Targeted class closure vs school closure. *BMC Infectious Diseases*, *14*(1), 695.

Génois, M, & Barrat, A. (2018). Can co-location be used as a proxy for face-to-face contacts? *EPJ Data Science*, *7*(1), 11.

Golovach, P.A., Heggernes, P., Lima, P.T., & Montealegre, P. (2020). Finding connected secluded subgraphs. *Journal of Computer and System Sciences*, 113, 101–124.

Himmel, A.-S., Molter, H., Niedermeier, R., & Sorge, M. (2017). Adapting the Bron-Kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining*, *7*(1), 35:1–35:16.

Holme, P., & Saramäki, J. (2019). *Temporal network theory*. Springer.

Håstad, J. (1999). Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, *182*(1), 105–142.

Hüffner, F., Komusiewicz, C., Moser, H., & Niedermeier, R. (2009). Isolation concepts for clique enumeration: Comparison and computational experiments. *Theoretical computer science*, *410*(52), 5384–5397.

Ito, H., & Iwama, K. (2009). Enumeration of isolated cliques and pseudo-cliques. *ACM Transactions on Algorithms*, *5*(4), 40:1–40:21.

Karp, R.M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations* (pp. 85–103). Springer.

Komusiewicz, C., Hüffner, F., Moser, H., & Niedermeier, R. (2009). Isolation concepts for efficiently enumerating dense subgraphs. *Theoretical Computer Science*, *410*(38-40), 3640–3654.

Latapy, M., Viard, T., & Magnien, C. (2018). Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, *8*(1), 61:1–61:29.

Michail, O. (2016). An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, *12*(4), 239–280.

Molter, H., Niedermeier, R., & Renken, M. (2019). Enumerating isolated cliques in temporal networks. *Proceedings of the 8th international conference on complex networks and their applications* (pp. 519–531). SCI, vol. 882. Springer.

Niedermeier, R. (2006). *Invitation to fixed-parameter algorithms*. Oxford University Press.

Rossetti, G., & Cazabet, R. (2018). Community discovery in dynamic networks: a survey. *ACM Computing Surveys*, *51*(2), 1–37.

Stehlé, J., Voirin, N., Barrat, A., Cattuto, C., Isella, L., Pinton, J.-F., Quaggiotto, M., Van den Broeck, W., Régis, C., Lina, B., *et al.* . (2011). High-resolution measurements of face-to-face contact patterns in a primary school. *PLOS ONE*, *6*(8).

Tomita, E., Tanaka, A., & Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, *363*(1), 28–42.

Viard, T., Latapy, M., & Magnien, C. (2016). Computing maximal cliques in link streams. *Theoretical Computer Science*, *609*, 245–252.

Viard, T., Magnien, C., & Latapy, M. (2018). Enumerating maximal cliques in link streams with durations. *Information Processing Letters*, *133*, 44–48.

Cite this article: Molter H., Niedermeier R. and Renken M. (2021). Isolation concepts applied to temporal clique enumeration. *Network Science* **9**, S83–S105. https://doi.org/10.1017/nws.2020.38