Enumerating Maximal Cliques in Temporal Graphs

Anne-Sophie Himmel, Hendrik Molter, Rolf Niedermeier and Manuel Sorge
Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany,
anne-sophie.himmel@campus.tu-berlin.de,
{h.molter, rolf.niedermeier, manuel.sorge}@tu-berlin.de

May 13, 2016

Abstract

Dynamics of interactions play an increasingly important role in the analysis of complex networks. A modeling framework to capture this are temporal graphs. We focus on enumerating $\Delta$-cliques, an extension of the concept of cliques to temporal graphs: for a given time period $\Delta$, a $\Delta$-clique in a temporal graph is a set of vertices and a time interval such that all vertices interact with each other at least after every $\Delta$ time steps within the time interval. Viard, Latapy, and Magnien [ASONAM 2015] proposed a greedy algorithm for enumerating all maximal $\Delta$-cliques in temporal graphs. In contrast to this approach, we adapt to the temporal setting the Bron-Kerbosch algorithm—an efficient, recursive backtracking algorithm which enumerates all maximal cliques in static graphs. We obtain encouraging results both in theory (concerning worst-case time analysis based on the parameter “$\Delta$-slice degeneracy” of the underlying graph) as well as in practice with experiments on real-world data. The latter culminates in a significant improvement for most interesting $\Delta$-values concerning running time in comparison with the algorithm of Viard, Latapy, and Magnien (typically two orders of magnitude).

1 Introduction

Network analysis is one of the main pillars of data science. Focusing on networks that are modeled by undirected graphs, a fundamental primitive is the identification of complete subgraphs, that is, cliques. This is particularly true in the context of detecting communities in social networks. Finding a maximum-cardinality clique in a graph is a classic NP-hard problem, so exponential worst-case running time seems unavoidable. Moreover, often one wants to solve the more general task of not only finding one maximum-cardinality clique but to enumerate all maximal cliques. Their number can be exponential in the graph size. The famous Bron-Kerbosch algorithm ("Algorithm 457" in Communications of the ACM 1973 [3]) addresses this task and still today forms the basis for the best (practical) algorithms to enumerate all maximal cliques in static graphs [4]. However, to realistically model many real-world phenomena in social and other network structures, one has to take into account the dynamics of the modeled system of interactions between entities, leading to so-called temporal networks. In a nutshell, compared to the standard static networks, in our model of temporal networks interactions (edges) can change over time (while the vertex set remains static). Indeed, as Nicosia et al. [18] pointed out, in many real-world systems the interactions among entities are rarely persistent over time and the non-temporal interpretation is an “oversimplifying approximation”. In this work, using a standard model with time-stamped edges and building on and improving previous work
for enumerating maximal ∆-cliques in temporal graphs \([24, 25]\), we present an adaption of the framework of Bron-Kerbosch to temporal graphs. To this end, we overcome several conceptual hurdles and propose a temporal version of the Bron-Kerbosch algorithm as a new standard for efficient enumeration of maximal cliques in temporal graphs.

1.1 Related Work

Our work relates to several lines of research. First, enumerating ∆-cliques in temporal graphs generalizes the enumeration of maximal cliques in standard graphs, this being subject to many different algorithmic approaches (sometimes also exploiting specific properties such as the “degree of isolation” of the cliques searched for) \([3, 4, 6, 11, 13, 21]\). Indeed, clique finding is a special case of dense subgraph detection. Second, more recently, mining dynamic or temporal networks for periodic interactions \([14]\) or preserving structures \([22]\) (in particular, this may include cliques as a very fundamental pattern) has gained increasing attention. Our work is directly motivated by the study of Viard et al. \([24, 25]\) who introduced the concept of ∆-cliques and provided a corresponding enumeration algorithm for ∆-cliques. In fact, following one of the their concluding remarks on future research possibilities, we adapt the Bron-Kerbosch algorithm to the temporal setting, thereby clearly outperforming their greedy-based approach.

1.2 Results and Organization

Our main contribution is to adapt the Bron-Kerbosch recursive backtracking algorithm for clique enumeration in static graphs to temporal graphs. In this way, we achieve a significant speedup for most interesting values for ∆ (typically two orders of magnitude) when compared to a previous algorithm due to Viard et al. \([24, 25]\) which is based on a greedy approach. We also contribute a theoretical running time analysis of our Bron-Kerbosch adaption employing the framework of parameterized complexity analysis. The analysis is based on the parameter “∆-slice degeneracy” which we introduce, an adaption of the degeneracy parameter that is frequently used in static graphs as a measure for sparsity. This extends results concerning the static Bron-Kerbosch algorithm \([1]\). A particular feature to achieve high efficiency of the standard Bron-Kerbosch algorithm is the use of pivoting, a procedure to reduce the number of recursive calls of the Bron-Kerbosch algorithm. We show how to define this and make it work in the temporal setting, where it becomes a significantly more delicate issue than in the static case. In summary, we propose our dynamic version of the Bron-Kerbosch approach as a current standard for enumerating maximal cliques in temporal graphs.

The paper is organized as follows. In Section 2 we introduce all main definitions and notations used in this paper. In addition, we give a description of the original Bron-Kerbosch algorithm as well as two extensions: pivoting and a running time upper bound using the degeneracy of the input graph. In Section 3 we propose an adaption of the Bron-Kerbosch algorithm to enumerate all maximal ∆-cliques in a temporal graph, prove the correctness of the algorithm and give a running time upper bound. Furthermore, we adapt the idea of pivoting to the temporal setting. In Section 4 we adapt the concept of degeneracy to the temporal setting and give an improved running time bound for enumerating all maximal ∆-cliques. In Section 5 we present the main results of the experiments on real-world data sets. We measure the ∆-slice degeneracy of real-world temporal graphs, we study the efficiency of our algorithm, and compare its running time to the algorithm of Viard et al. \([24]\), showing a significant performance increase due to our Bron-Kerbosch approach. We conclude in Section 6.
2 Preliminaries

In this section we introduce the most important notations and definitions used throughout this article.

2.1 Graph-Theoretic Concepts

In the following we provide definitions of adaptations to the temporal setting for central graph-theoretic concepts.

2.1.1 Temporal Graphs

The goal of the concept of temporal graphs, also referred to as time-varying graphs [18], temporal networks [8], or link streams [24], is to capture changes in a graph that occur over time. In this work, we use the well-established model where each edge is provided with a time stamp [2, 8, 24]. Assuming discrete time steps, this is equivalent to a sequence of static graphs over a fixed set of vertices [3, 16, 17]. Formally, the model is defined as follows.

Definition 1 (Temporal Graph). A temporal graph $G = (V, E, T)$ is defined as a triple consisting of a set of vertices $V$, a set of time-edges $E \subseteq (V \times V) \times T$, and a time interval $T = [\alpha, \omega]$, where $\alpha, \omega \in \mathbb{N}$, $T \subseteq \mathbb{N}$ and $\omega - \alpha$ is the lifetime of the temporal graph $G$.

2.1.2 $\Delta$-Cliques

A straightforward adaptation of a clique to the temporal setting would be to additionally assign a lifetime $I = [a, b]$ to it, that is the largest time interval such that the clique exists in each time step in said interval. However, this model is often too restrictive for real-world data. The constraint of each pair of vertices being connected in each time step can be relaxed by introducing an additional parameter, $\Delta$, quantifying how many time steps may be skipped between two connections of any vertex pair. These so-called $\Delta$-cliques were first introduced by Viard et al. [24, 25] and can formally be defined as follows.

Definition 2 ($\Delta$-Clique). Let $\Delta \in \mathbb{N}$. A $\Delta$-clique in a temporal graph $G = (V, E, T)$ is a tuple $C = (X, I = [a, b])$ with $X \subseteq V$, $a - b \geq \Delta$, and $I \subseteq T$ such that for all $\tau \in [a, b - \Delta]$ and for all $v, w \in X$ with $v \neq w$ there exists a $(\{v, w\}, t) \in E$ with $t \in [\tau, \tau + \Delta]$.

Note that we also need to adapt the notion of maximality to the temporal setting. We call a $\Delta$-clique $C = (X, I)$ vertex-maximal if we cannot add any vertex to $X$ without having to decrease the clique’s lifetime $I$. We say that a $\Delta$-clique is time-maximal if we cannot increase the lifetime $I$ without having to remove vertices from $X$. We call a $\Delta$-clique maximal if it is both vertex-maximal and time-maximal.

2.1.3 $\Delta$-Neighborhood

We define a tuple $(v, I = [a, b])$ with $v \in V$ and $I \subseteq T$ as a vertex-interval pair of a temporal graph. We need this definition to adapt the notion of a neighborhood of a vertex to temporal graphs. For a vertex $v \in V$ and a time interval $I \subseteq T$ in a temporal graph, we define the $\Delta$-neighborhood $N^\Delta(v, I)$ as the set of all vertex-interval pairs $(w, I' = [a', b'])$ with the property that for every $\tau \in [a', b' - \Delta]$ at least one edge $(\{v, w\}, t) \in E$ with $t \in [\tau, \tau + \Delta]$ exists. Furthermore, we require that $b' - a' \geq \Delta$, $I' \subseteq I$ and $I'$ is maximal, that is, there is no time interval $I'' \subseteq I$ with $I' \subset I''$ satisfying the properties above. In Figure 1 we visualize the concept of $\Delta$-neighborhood and $\Delta$-clique in a temporal graph.
Figure 1: $\Delta$-Neighborhoods and a $\Delta$-Clique of a temporal graph, $\Delta = 2$. $T$ is the lifetime of the graph. The elements of the $\Delta$-neighborhoods are colored in yellow and green, respectively.

Algorithm 1 Enumerating all Maximal Cliques

1: function BronKerbosch($P, R, X$)
2: if $P \cup X = \emptyset$ then
3: add $R$ to the solution
4: end if
5: for $v \in P$ do
6: BronKerbosch($P \cap N(v), R \cup \{v\}, X \cap N(v)$)
7: $P \leftarrow P \setminus \{v\}$
8: $X \leftarrow X \cup \{v\}$
9: end for
10: end function

2.1.4 Temporal Membership

Let $X$ be a set of vertex-interval pairs. The relation $(v, I) \in X$ expresses that a vertex-interval pair $(v, I') \in X$ with $I \subseteq I'$ exists. If $(w, I') \in N^\Delta(v, I)$, then we say that $w$ is a $\Delta$-neighbor of $v$ during the time interval $I'$.

2.1.5 $\Delta$-Cut

Let $X$ and $Y$ be sets of vertex-interval pairs. The $\Delta$-cut $X \cap Y$ includes all pairs $(v, I = [a, b])$ such that $(v, I) \in X$ and $(v, I) \in Y$, as well as $b - a \geq \Delta$ and $I$ is maximal, that is, there is no $J$ with $I \subset J$ and $J \subseteq I'$ and $J \subseteq I''$ such that $(v, I') \in X$ and $(v, I'') \in Y$ for some $I'$ and $I''$.

2.2 Bron-Kerbosch Algorithm

The Bron-Kerbosch algorithm is a widely used algorithm for enumerating all maximal cliques in undirected, static graphs. It is a recursive backtracking algorithm, which is easy to implement and has often been shown to be more efficient in practice than alternative algorithms.

The Bron-Kerbosch algorithm (Algorithm 1) receives three disjoint vertex-sets as an input: $P$, $R$, and $X$. The set $R$ induces a clique and $P \cup X$ is the set of all vertices which are adjacent to every vertex in $R$. Each vertex in $P \cup X$ is a witness that the clique $R$ is not maximal yet. The set $P$ contains the vertices that have not been considered yet whereas the set $X$ includes all vertices that have already been considered in earlier steps. This set is used to
Algorithm 2 Enumerating all Maximal $\Delta$-Clique

\begin{algorithm}
\begin{algorithmic}[1]
\Function{BronKerboschDelta}{$P, R = (C, I), X$}
\State if \{ $(w, I) \mid (w, I) \in P \cup X$ \} = $\emptyset$ then
\State add $R$ to the solution
\EndIf
\For{$(v, I') \in P$}
\State $R' \leftarrow (C \cup \{v\}, I')$
\State $P' \leftarrow P \cap N^\Delta(v, I')$
\State $X' \leftarrow X \cap N^\Delta(v, I')$
\State \Call{BronKerboschDelta}{$P', R', X'$}
\EndFor
\State $P \leftarrow P \setminus \{(v, I')\}$
\State $X \leftarrow X \cup \{(v, I')\}$
\EndFunction
\end{algorithmic}
\end{algorithm}

avoid enumerating maximal cliques multiple times. In each call, the algorithm checks whether the given clique $R$ is maximal or not. If $P \cup X = \emptyset$, then there are no vertices that can be added to the clique. Therefore, the clique is maximal and can be added to the solution. For a graph $G = (V, E)$ the algorithm is initially called with $P = V$ and $R = X = \emptyset$.

2.2.1 Pivoting

Bron and Kerbosch \[3\] introduced a method to increase the efficiency of the basic algorithm by choosing a pivot element to decrease the number of recursive calls. It is based on the observation that for any vertex $u \in P \cup X$ either $u$ itself or one of its non-neighbors must be contained in any maximal clique containing $R$. Hence, choosing an arbitrary pivot element $u \in P \cup X$ and iterating just over $u$ and all its non-neighbors decreases the number of recursive calls in the for-loop that lead to non-maximal cliques. Tomita et al. \[21\] have shown that if $u$ is chosen from $P \cup X$ such that $u$ has the most neighbors in $P$, then all maximal cliques of a graph $G = (V, E)$ can be enumerated in $O(3^{\frac{3|V|}{3}})$ time.

2.2.2 Degeneracy

A static graph $G$ has degeneracy $d \in \mathbb{N}$ if $d$ is smallest-possible such that each subgraph $G'$ of $G$ contains a vertex $v$ with $\deg(v) \leq d$. If the degeneracy of a graph is $d$, then the maximal clique size of the graph is at most $d + 1$: If there is a clique of size at least $d + 2$, then the vertices of this clique would form a subgraph in which every vertex $v$ of the clique has $\deg(v) \geq d + 1$. Eppstein et al. \[4\] showed that Algorithm 1 can be modified such that all maximal cliques of a graph with degeneracy $d$ can be enumerated in time $O(d \cdot |V| \cdot 3^{d/3})$.

3 Bron-Kerbosch for Temporal Graphs

We adapt the classic Bron-Kerbosch algorithm to the temporal setting to enumerate all $\Delta$-cliques, see Algorithm 2. The input of the algorithm consists of two sets $P$ and $X$ of vertex-interval pairs as well as a tuple $R = (C, I)$, where $C$ is a set of vertices and $I$ a time interval. We will show that $R$ is a time-maximal $\Delta$-clique. The sets $P$ and $X$ contain vertex-interval pairs that are in the $\Delta$-neighborhood of every vertex in $C$ during an interval $I' \subseteq I$. Particularly, $P \cup X$ includes all vertex-interval pairs $(v, I)$ for which $(C \cup \{v\}, I)$ is a time-maximal $\Delta$-clique. While each vertex-interval pair in $P$ still has to be combined with $R$ to ensure that every maximal $\Delta$-clique will be found, for every vertex-interval pair $(v, I') \in X$ every maximal $\Delta$-clique $(C', I'')$ with $C \cup \{v\} \subseteq C'$ and $I'' \subseteq I'$ has already been detected in earlier steps.
We show below that, if \( \{(w, I) \mid (w, I) \in P \cup X\} = \emptyset \), then there is no vertex \( v \) that forms a \( \Delta \)-clique together with \( C \) over the whole time interval \( I \). Consequently, \( R = (C, I) \) is a maximal \( \Delta \)-clique.

For a temporal graph \( G = (V, E, T) \) and a given time period \( \Delta \), the initial call for Algorithm \( 2 \) to enumerate all maximal \( \Delta \)-cliques in graph \( G \) is made with \( P = \{(v, T) \mid v \in V\} \), \( R = (\emptyset, T) \) and \( X = \emptyset \). In the remainder of this document we will always assume that BRONKERBOSSCHDELTA is initially called with those inputs.

### 3.1 Analysis

In the following, we prove correctness of the algorithm and analyze its running time. We start with arguing that the sets \( P \) and \( X \) behave as claimed.

**Lemma 1.** For each recursive call of BRONKERBOSSCHDELTA with \( R = (C, I) \) and \( C \neq \emptyset \) it holds that \( P \cup X = \bigcap_{v \in C} N^\Delta(v, I) \).

**Proof.** We prove this by induction on the recursion depth. In the initial call we have that \( C = \emptyset \). In each iteration of the first call we have that \( P \cup X = \{(v, T) \mid v \in V\} \) since whenever a vertex is removed from \( P \) it is added to \( X \) and initially \( P = \{(v, T) \mid v \in V\} \). So for every recursive call of BRONKERBOSSCHDELTA with \( R' = (C', I') \), \( P' \), and \( X' \), and \( C' = \{v\} \) for some vertex \( v \) we have that \( P' = P \cap N^\Delta(v, I') \) and \( X' = X \cap N^\Delta(v, I') \). Hence, we get

\[
P' \cup X' = \{(v, T) \mid v \in V\} \cap N^\Delta(v, I') = N^\Delta(v, I').
\]

Now assume that we are in a recursive call of BRONKERBOSSCHDELTA with \( R = (C, I) \), \( P \), and \( X \), and \(|C| > 1\). Let \((v, I') \in P \) be the vertex added to the \( \Delta \)-clique, that is, in the next recursive call we have that \( R' = (C', I') \), with \( C' = C \cup \{v\} \), and \( P' = P \cap N^\Delta(v, I') \) as well as \( X' = X \cap N^\Delta(v, I') \). Then,

\[
P' \cap X' = (P \cup X) \cap N^\Delta(v, I')
\]

\[
= \bigcap_{w \in C} N^\Delta(w, I) \cap N^\Delta(v, I')
\]

\[
= \bigcap_{w \in C'} N^\Delta(w, I').
\]

This concludes the proof. \( \square \)

Next, we show that the \( R \) behaves as claimed.

**Lemma 2.** In each recursive call of BRONKERBOSSCHDELTA \( R = (C, I) \) is a time-maximal \( \Delta \)-clique.

**Proof.** We show by induction on the recursion depth that \( R = (C, I) \) is a time-maximal \( \Delta \)-clique and that all vertex-interval pairs \( (v, I') \) in \( P \) are \( \Delta \)-neighbors during \( I' \) to all vertices in the \( \Delta \)-clique \( R \) and that \( I' \) is maximal under that property. The algorithm is initially called with \( R = (\emptyset, T) \), which is a trivial time-maximal \( \Delta \)-clique, and \( P = \{(v, T) \mid v \in V\} \), which fulfills the desired property since the initial \( \Delta \)-clique is empty and \( T \) is the maximal time interval.

In each recursion call BRONKERBOSSCHDELTA is called with \( (P \cap N^\Delta(v, I'), (C \cup \{v\}, I'), X \cap N^\Delta(v, I')) \) for some \((v, I') \in P \). By the induction hypothesis \( v \) is a \( \Delta \)-neighbor to all vertices in \( C \) during time interval \( I' \) and \( I' \) is maximal. It follows that \( (C \cup \{v\}, I') \) is a time-maximal \( \Delta \)-clique. Furthermore, each vertex-interval pair \((v', I'') \) in \( P \cap N^\Delta(v, I') \) is in the \( \Delta \)-neighborhood of each vertex-interval pair \((v'', I') \) with \( v'' \in C \cup \{v\} \), since it is both in \( P \) and hence in
the \( \Delta \)-neighborhood of each vertex in \( C \) and in \( N^\Delta(v, I') \). The maximality of \( I' \) follows from the fact that the \( \Delta \)-cut and \( \Delta \)-neighborhood operations preserve maximality of intervals by definition.

Now we can prove the correctness of the algorithm.

**Theorem 1** (Correctness). Let \( \mathcal{G} = (V, E, T) \) be a temporal graph. If \( \text{BronKerboschDelta}(P, R, X) \) is run on input \( (V \times \{T\}, \emptyset, \emptyset) \), then each maximal \( \Delta \)-clique \( R \) of \( \mathcal{G} \) is added to the solution.

**Proof.** Let \( R = (C, I) \) be a maximal \( \Delta \)-clique with \( |C| > 1 \). Assume for contradiction that this clique is not added to the solution. Let \( R' = (C', I') \) be a \( \Delta \)-clique with \( C' \subset C, |C| > 0 \), and \( I \subset I' \) that is maximal with respect to vertex inclusion and that occurs in a recursive call of \( \text{BronKerboschDelta} \). Such a \( \Delta \)-clique exists since \( \text{BronKerboschDelta} \) is initially called with \( R = (\emptyset, T) \) which fulfills these conditions. Consider the sets \( P' \) and \( X' \) corresponding to the recursive call of \( \text{BronKerboschDelta} \) with \( R' \). Lemma 1 implies that \( \exists (v, I'') \in P' \cup X' \) with \( v \in C \setminus C' \) and \( I \subset I'' \).

We first consider the case that \( (v, I'') \in P' \). Then the algorithm will eventually make a recursive call with \( R'' = (C' \cup \{v\}, I'') \).

Now consider the case that \( (v, I'') \in X' \). Here we have that the \( \Delta \)-clique \( R'' = (C' \cup \{v\}, I'') \) already occurred in a previous recursive call of \( \text{BronKerboschDelta} \).

Both cases are a contradiction to \( R' \) being maximal with respect to vertex inclusion among all \( \Delta \)-cliques that occur in recursive calls of \( \text{BronKerboschDelta} \). Hence, indeed, \( R = (C, I) \) is added to the solution.

Next, we analyze the running time of \( \text{BronKerboschDelta} \). We start with the following observation.

**Lemma 3.** For each time-maximal \( \Delta \)-clique \( R \) of a temporal graph \( \mathcal{G} = (V, E, T) \), there is at most one recursive call of \( \text{BronKerboschDelta} \) with \( R \) as an input.

**Proof.** Assume that there are two recursive calls of \( \text{BronKerboschDelta} \) with the same \( R = (C, I) \). Let \( R' = (C', I') \), with \( C' \subset C \) and \( I \subset I' \) occur in the recursive call corresponding to their closest common ancestor in the recursion tree. Let \( v \in C \setminus C' \). We know that \( (v, I'') \in P' \) for some \( I \subset I'' \subset I' \), otherwise \( R' \) would not be an ancestor of \( R \). After \( \text{BronKerboschDelta} \) is recursively called with \( R'' = (C' \cup \{v\}, I'') \), \( (v, I'') \) is removed from \( P' \) and added to \( X' \), hence there cannot be any second recursive call of \( \text{BronKerboschDelta} \) with \( R'' = (C' \cup \{v\}, I'') \).

Now we bound the running time for computing a \( \Delta \)-cut.

**Lemma 4.** Let \( X \) and \( Y \) be two sets of vertex-interval pairs such that for every \( (v, I) \in X \) and \( (v, I') \in Y \) and also every \( (v, I) \in Y \) and \( (v, I') \in Y \), we have that \( |I \cap I'| < \Delta \). Furthermore, assume that \( X \) and \( Y \) are sorted lexicographically by first the vertex and then the starting time of the interval. Then the \( \Delta \)-cut \( X \cap Y \) can be computed in \( O(\max(|X|, |Y|)) \) time such that it is also sorted lexicographically by first the vertex and then the starting time of the interval.

**Proof Sketch.** The \( \Delta \)-cut \( X \cap Y \) of two sets of vertex-interval pairs \( X \) and \( Y \) can be computed in the following way.

- For each vertex \( v \), first select vertex-interval pairs \( (v, I) \) and \( (v, I') \) from \( X \) and \( Y \), respectively.
- If \( |I \cap I'| > \Delta \), then add \( (v, I \cap I') \) to the \( \Delta \)-cut. If the endpoint of \( I' \) is earlier than the endpoint of \( I \), then proceed with the next vertex-interval in \( Y \), otherwise proceed with the next vertex in \( X \). Repeat until all vertex-interval pairs containing vertex \( v \) are processed.
Note that the intervals for each vertex \( v \) are added to the \( \Delta \)-cut in order of their starting time. It is easy to verify that each vertex-interval pair is only touched once and hence the running time is in \( O(\max(|X|, |Y|)) \).

Lemmas 3 and 4 allow us to bound the running time of \textsc{BronKerboschDelta} depending on the number of different time-maximal \( \Delta \)-cliques of the input graph.

**Lemma 5.** Given a temporal graph \( G = (V, E, T) \) with \( x \) different time-maximal \( \Delta \)-cliques. Then \textsc{BronKerboschDelta} enumerates all maximal \( \Delta \)-cliques in \( O(x \cdot |E| + |E| \cdot |T|) \) time.

**Proof.** We assume that all edges of the temporal graph are sorted by their time stamp. Note that this can be done in a preprocessing step in \( O(|E| \cdot |T|) \) time using counting sort. Furthermore, we assume that for each vertex \( v \), the \( \Delta \)-neighborhood \( N^\Delta(v, T) \) is given. These neighborhoods can be precomputed in \( O(|E|) \) time, assuming that the edges are sorted by their time stamps.

By Lemma 3 we know that for each time-maximal \( \Delta \)-clique there is at most one recursive call of \textsc{BronKerboschDelta}. By charging the computation of \( P' \), \( R' \), and \( X' \) to the corresponding recursive call, for each recursive call we compute a constant number of \( \Delta \)-neighborhoods and \( \Delta \)-cuts. The size of the sets \( P \), \( X \), and any \( \Delta \)-neighborhood is upper-bounded by \(|E|\) and each of these sets has the property that for every \((v, I)\) and \((v', I')\) out of the same set we have that \(|I \cap I'| < \Delta\). Given \( N^\Delta(v, T) \), \( N^\Delta(v, I) \) can be computed in time \( O(|E|) \) for any \( I \) and by Lemma 4 a \( \Delta \)-cut can be computed in \( O(|E|) \) time. Hence, all maximal \( \Delta \)-cliques can be enumerated in \( O(x \cdot |E| + |E| \cdot |T|) \) time.

Now we can bound the overall running time of \textsc{BronKerboschDelta}.

**Theorem 2.** Given \( G = (V, E, T) \) as input, the running time of \textsc{BronKerboschDelta} is in \( O(2^{|V|} \cdot |T| \cdot |E|) \).

**Proof.** Note that the vertex set of each maximal \( \Delta \)-clique induces a static clique in the static graph \( G \) underlying \( G \) that has an edge between two vertices if there is a time-edge in \( G \) between these vertices at some time step. Furthermore, for each clique in \( G \), there are at most \(|T|\) maximal \( \Delta \)-cliques because their time intervals are pairwise not contained in one-another. Hence, the number of time-maximal \( \Delta \)-cliques of any temporal graph is bounded by \( 2^{|V|} \cdot |T| \cdot |E| \). By Lemma 3 we get an overall running time in \( O(2^{|V|} \cdot |T| \cdot |E|) \).

### 3.2 Pivoting

Recall that the idea of pivoting in the Bron-Kerbosch algorithm for non-temporal graphs is based on the observation that for any vertex \( u \in P \cup X \)—the pivot element—either \( u \) itself or one of its non-neighbors must be contained in any maximal clique containing \( R \). This observation also holds for maximal \( \Delta \)-cliques in temporal graphs: for any \((v_p, I_p) \in P \cup X \) in any maximal \( \Delta \)-clique \( R_{\text{max}} = (C_{\text{max}}, I_{\text{max}}) \) with \( C \subseteq C_{\text{max}} \) and \( I_{\text{max}} \subseteq I_p \subseteq I \) either the vertex \( v_p \) or one vertex \( w \) which is not a \( \Delta \) neighbor of \( v_p \) during the time \( I_{\text{max}} \), that is \((w, I_{\text{max}}) \in N^\Delta(v_p, I_p)\), must be contained in \( C_{\text{max}} \).

By choosing a pivot element \((v_p, I_p) \in X \cup P \) we only have to iterate over all elements in \( P \) which are not in the \( \Delta \)-neighborhood of the pivot element. In other words, we do not have to make a recursive call for any \((w, I') \in P \) which holds \((w, I') \in N^\Delta(v_p, I_p)\). We omit a formal proof showing that pivoting preserves the correctness of the algorithm.

An optimal pivot element is chosen in such a way that it minimizes the number of recursive calls. It is the element in the set \( P \cup X \) of which the most elements in \( P \) are in its \( \Delta \)-neighborhood. We have seen that the whole procedure is quite similar to pivoting in the basic Bron-Kerbosch algorithm but with one difference: we are able to choose more than one
pivot element. The only condition that has to be satisfied is that the time intervals of the pivot elements cannot overlap.

For each $\Delta$-clique $R = (C, I)$ in a node of the recursion tree, choosing a pivot element $(v_p, I_p) \in P \cup X$ only affects maximal $\Delta$-cliques $R_{\max} = (C_{\max}, I_{\max})$ fulfilling $I_{\max} \subseteq I_p$. Moreover, for all elements $(w, I') \in P$ satisfying $(w, I') \in N^\Delta(v_p, I_p)$ it holds $I' \subseteq I_p$. Consequently, a further pivot element $(v'_p, I'_p) \in P \cup X$ fulfilling that $I'_p$ does not overlap with $I_p$ neither interferes with the considered maximal $\Delta$-cliques nor with the vertex-interval pairs in $P$ that are in the $\Delta$-neighborhood of the pivot element $(v_p, I_p)$. The problem of finding the optimal set of pivot elements in $P \cup X$ can be formulated as a weighted interval scheduling maximization problem:

**Definition 3** ($\Delta$-slice degeneracy). A temporal graph $G = (V, E, T)$ has $\Delta$-slice degeneracy $d$ if for all $t \in T$ we have that the graph $G_t = (V, E_t)$, where $E_t = \{(v, w) \mid (u, w, t') \in E \text{ for some } t' \in [t, t + \Delta]\}$, has degeneracy at most $d$.

Using the parameter $\Delta$-slice degeneracy, we can bound the number of time-maximal $\Delta$-cliques of a temporal graph.

**Lemma 6.** Let $G = (V, E, T)$ be a temporal graph with $\Delta$-slice degeneracy $d$, then the number of time-maximal $\Delta$-cliques in $G$ is at most $3^\frac{d}{d+1} \cdot |V| \cdot |T|$.

**Proof.** Let $G = (V, E, T)$ be a temporal graph with $\Delta$-slice degeneracy $d$. Then we call the graph $G_t = (V, E_t)$, where $E_t = \{(v, w) \mid (u, w, t') \in E \text{ for some } t' \in [t, t + \Delta]\}$ a $\Delta$-slice of $G$ at time $t$. The vertex set of each time-maximal $\Delta$-clique which starts at time $t$ is also a clique in $G_t$, otherwise there would be two vertices which are disconnected for more than $\Delta$ time-steps. Since $G_t$ has degeneracy at most $d$ the number of maximal cliques of $G_t$ is upper-bounded by $3^\frac{d}{d+1} \cdot |V|$ [4]. Furthermore the maximal size of a clique is bounded by $d + 1$. Hence, the total number of cliques is bounded by $3^\frac{d}{d+1} \cdot |V|$. Note that for each of those cliques we have at most one time-maximal $\Delta$-clique starting at time $t$. Hence, the total number of $\Delta$-cliques is bounded by $3^\frac{d}{d+1} \cdot |V| \cdot |T|$.

\[ \square \]
Lemma 6 now implies a more precise running time bound.

**Theorem 3.** Let \(d\) be the \(\Delta\)-slice degeneracy of a temporal graph \(G = (V, E, T)\). BRONKERBOSCHDELTA enumerates all \(\Delta\)-cliques of \(G\) in time \(O(3^d \cdot 2^d \cdot |V| \cdot |T| \cdot |E|)\).

**Proof.** By Lemma 6 we know that the number of time-maximal \(\Delta\)-cliques in a temporal graph with \(\Delta\)-slice degeneracy \(d\) is at most \(3^d \cdot 2^{d+1} \cdot |V| \cdot |T|\). Hence, by Lemma 6 we get an overall running time in \(O(3^d \cdot 2^d \cdot |V| \cdot |T| \cdot |E|)\).

Note that Theorem 3 means that enumerating all maximal \(\Delta\)-cliques is fixed-parameter tractable with respect to the parameter \(\Delta\)-slice degeneracy. Hence, while NP-hard in general, the problem can be solved efficiently if the \(\Delta\)-slice degeneracy of the input graph is small.

## 5 Experimental Results

We now report on the efficiency of BRONKERBOSCHDELTA on several real-world temporal graphs and their \(\Delta\)-slice degeneracy.

### 5.1 Setup and Statistics

We first explain the implementation, used reference algorithm, data sets, and the chosen values of \(\Delta\).

**Implementation** We implemented BRONKERBOSCHDELTA and an algorithm to compute the \(\Delta\)-slice degeneracy in Python 2.7.11 and carried out experiments on an Intel Xeon E5-1620 computer with 4 cores clocked at 3.6 GHz and 64 GB RAM. We did not utilize the parallel-processing capabilities although it should be easy to achieve almost linear speed-up with growing number of cores due to simple nature of BRONKERBOSCHDELTA. The operating system was Debian GNU/Linux 6.0. We compare BRONKERBOSCHDELTA with the algorithm by Viard, Latapy and Magnien [25] which was also implemented in Python. We modified their source code by removing the (extensive) text output in order to avoid possible output bottlenecks. We call their algorithm Algorithm VLM below.

**Data Sets** We used several freely available real-world temporal graphs including

- an internet router communication network (as733 [15]),
- an email communication network (karlsruhe [7]),
- social-network communication network (facebook-like [19]), and
- physical-proximity networks between high school students (highschool-2011, highschool-2012, highschool-2013 [1, 6, 20]),
- between patients and health-care workers (hospital-ward [23]),
- conference attendees of ACM Hypertext 2009 (hypertext [10]),
- attendees of the Infectious SocioPatterns event (infectious [10]), and
- between children and teachers in a primary school (primaryschool [20]).

Table 1 contains important statistics.

---

1Code available at http://fpt.akt.tu-berlin.de/temporalcliques/ .
2Available at https://github.com/JordanV/delta-cliques/ .
3Available at http://www.sociopatterns.org/datasets/ .
Table 1: Statistics for the data sets used in our experiments.

| Instance          | Vertices | Edges   | Resolution | Lifetime (s) |
|-------------------|----------|---------|------------|--------------|
| as733             | 7,716    | 11,410,810 | 1d         | 67,824,000   |
| facebook-like     | 1,899    | 59,835  | 1s         | 16,736,181   |
| highschool-2011   | 126      | 28,560  | 20s        | 272,330      |
| highschool-2012   | 180      | 45,047  | 20s        | 729,500      |
| highschool-2013   | 327      | 188,508 | 20s        | 363,560      |
| hospital-ward     | 75       | 32,424  | 20s        | 347,500      |
| infectious         | 113      | 20,818  | 20s        | 212,340      |
| karlsruhe         | 1,870    | 461,661 | 1s         | 123,837,267  |
| primaryschool     | 242      | 125,773 | 20s        | 116,900      |

Table 2: Classical degeneracy and ∆-slice degeneracy.

| Instance     | Classical | ∆ = 0 | ∼ 5^3 | ∼ 5^5 | ∼ 5^7 | ∼ 5^9 |
|--------------|-----------|-------|-------|-------|-------|-------|
| as733        | 24        | 13    | 13    | 14    | 15    | 24    |
| facebook-like| 20        | 1     | 3     | 6     | 19    |       |
| highschool-2012 | 18    | 4     | 5     | 6     | 12    |       |
| hospital-ward | 22    | 4     | 6     | 11    | 18    |       |
| hypertext    | 28        | 6     | 7     | 8     | 22    |       |
| infectious    | 18        | 4     | 9     | 18    | 18    | 18    |
| karlsruhe    | 33        | 2     | 6     | 9     | 17    | 32    |
| primaryschool | 47        | 4     | 4     | 10    | 31    |       |

Chosen values of ∆  In order to limit the influence of different time scales in the data and, hence, to make running times more comparable between instances as well as to be able to present the results in a unified way, we chose the ∆ values as follows. We decided on a reference point on the average number of edges per time step and on a set of ∆ values for this reference point. For each considered instance we then scaled the reference ∆ values by the quotient of the average number of edges per time step in the reference point divided by the average number of edges per time step in the instance.

As the reference point we chose the rate of 1/5 edges per time step; this value was chosen for convenience within the interval of average edge-appearance rates in the studied data sets (see Table 1). Since, intuitively, the ∆ values of interest in practice increase exponentially, we chose as ∆ values for the reference point 0 and 5^i for i ∈ N. As mentioned, for each instance, these values are then multiplied by (1/5)/(|E|/|T|) = |T|/(5|E|). For example, for highschool-2012 we obtain the ∆ values {0, 80, 404, 2024, 10121, 50606, 253034, . . .} (for reference, recall that each time step in highschool-2012 corresponds to one second). In figures, we refer to each scaled value of ∆ by ∼ 5^i for some concrete i.

∆-slice degeneracy  The ∆-slice degeneracy for our set of instances is shown in Table 2 as well as the classical degeneracy of the underlying static graph which has an edge whenever there is an edge at some time step in the temporal graph. Clearly, as the value of ∆ increases, the ∆-slice degeneracy approaches, and is upper-bounded by the classical degeneracy. The classical degeneracy of our instances is small in comparison with the size of the graph. This falls in line with the analysis by Eppstein, Löffler, and Strash [4] for many real-world graphs. Moreover, for
many practically interesting values of $\Delta$ the $\Delta$-slice degeneracy is still much smaller. For example, as mentioned above, the scaled value of $\Delta$ corresponding to $5^3$ for highschool-2012 equals 2204 time steps (seconds) and the corresponding $\Delta$-slice degeneracy is 5.

5.2 Results and Running Times

We now study the efficiency of BronKerboschDelta, evaluate pivoting strategies, and compare the best one to Algorithm VLM.

Pivoting Generally we observe that pivoting plays a negligible role when $\Delta$ is small compared to the overall lifetime of the graph, that is, when $\Delta$ is less than roughly one third of the lifetime. In this case, pivoting has almost no effect on the running time and the number of recursive calls. However, for larger values of $\Delta$, pivoting makes a clear difference.

We tested five strategies for selecting a set of pivots from $P$ in BronKerboschDelta:

1A) a single arbitrary pivot,
1G) a single pivot maximizing the number of elements removed from $P$,
MA) an arbitrary maximal set of pivots,
MG) a maximal set of pivots, picked one by one according to the maximum number of further elements removed from $P$, and
MM) a (maximal) set of pivots which maximizes the number of elements removed from $P$.

Herein, we say that a set of pivots is maximal if the corresponding intervals do not overlap. Clearly, each strategy has its own trade-off between the time needed to compute the pivots and the possible reduction in recursive calls.

Running times are given for highschool-2012 in Figure 2 with $\Delta \in [15000, 725000]$. For $\Delta \leq 15000$ there is no appreciable difference between the pivoting strategies. In terms of relative difference between pivoting strategies, highschool-2012 seems to be a representative example. Strategies 1G and MG seem to be the best options: they do not incur much overhead compared to no pivoting for small $\Delta$ and yield strong running time improvements for larger $\Delta$. In comparison to no pivoting, strategy 1G and MG achieve a 60% reduction in recursive calls for $\Delta$ values of around $7 \cdot 10^6$ in highschool-2012. Since the running times of strategy 1G and MG are so close we conclude that in most cases there is only one important pivot that should be selected. Finally, it is interesting that maximizing the overall number of elements removed from $P$ via

![Figure 2: Running time for different pivoting strategies on highschool-2012.](image)
Table 3: \( \Delta \)-clique statistics and running times: \(|\mathcal{C}|\) denotes the number of maximal \( \Delta \)-cliques (excluding cliques containing at most one vertex), \( s \) denotes the maximum \( \Delta \)-clique size, \( \ell \) the maximum \( \Delta \)-clique lifetime divided by \( 10^5 \), \( t_2 \) and \( t_{VLM} \) denote the running time in seconds of \textsc{BronKerboschDelta} and Algorithm VLM, respectively. Empty cells represent an exceeded running time limit of one hour.

| Instance        | \( |\mathcal{C}| \) | \( s \) | \( \ell \) | \( t_2 \) | \( t_{VLM} \) | \( |\mathcal{C}| \) | \( s \) | \( \ell \) | \( t_2 \) | \( t_{VLM} \) |
|-----------------|---------------------|--------|----------|--------|-------------|---------------------|--------|----------|--------|-------------|
| facebook-like   | 61,648              | 2      | 1,674    | 160    | 10          | 33,876              | 4      | 1,675    | 65     | 1,651       |
| highschool-2011 | 26,510              | 5      | 27       | 121    | 6           | 7,394               | 7      | 27       | 5      | 168         |
| highschool-2012 | 42,285              | 5      | 73       | 231    | 10          | 9,501               | 6      | 73       | 8      | 213         |
| highschool-2013 | 172,362             | 5      | 36       | 1,804  | 118         | 57,121              | 6      | 36       | 140    | 2,058       |
| hospital-ward   | 27,910              | 5      | 35       | 360    | 12          | 8,694               | 7      | 35       | 14     | 199         |
| hypertext       | 19,150              | 6      | 21       | 82     | 5           | 6,345               | 7      | 21       | 7      | 96          |
| infectious      | 349,787             | 5      | 695      | 1,493  | 3,006       | 134,787             | 9      | 695      | 1,156  | 163,162     |
| karlsruhe       | 1,337               |        |          |        |             | 235,684             | 9      | 12,417   | 1,163  |             |
| primaryschool   | 107,121             | 5      | 12       | 931    | 140         | 83,314              | 9      | 12       | 78     | 508,430     |

the pivot set (strategy MM) results in slightly worse running times and slightly larger numbers of recursive calls. All remaining experiments were carried out with strategy 1G.

Running Times and Comparison with Algorithm VLM We performed experiments with \textsc{BronKerboschDelta} and Algorithm VLM for \( \Delta = 0 \) and \( \sim 5^3, 5^5 \) (where the lifetime allowed such values of \( \Delta \)). An excerpt of the results is given in Table 3. Clearly, larger instances with more vertices and/or edges demand a longer running time. However, even large instances like infectious can still be solved within one hour.

From our theoretical results in Section 3 we expect that the running time of \textsc{BronKerboschDelta} increases exponentially with growing \( \Delta \)-slice degeneracy. As the \( \Delta \)-slice degeneracy grows very slowly with increasing \( \Delta \) (see Table 2), we expect a corresponding moderate growth in running time with respect to \( \Delta \). For larger \( \Delta \), this is consistent with the experimental results, as we show in Figure 2 and Table 3. However, for (very) small \( \Delta \) we observe an initial spike in the running time (and number of \( \Delta \)-cliques) which then subsides (Figure 3).

A possible explanation for this spike is that, for small \( \Delta \), the \( \Delta \)-neighborhood of many vertices becomes very fragmented, leading to large sets of \( P \) in the algorithm (although the size of \( P \) is still linear in the input size for constant \( \Delta \)-slice degeneracy). Furthermore, if \( \Delta \) is small, many singleton edges may form maximal \( \Delta \)-cliques themselves. These \( \Delta \)-cliques then get taken up into larger maximal \( \Delta \)-cliques when \( \Delta \) increases, which decreases the number \( \Delta \)-cliques and running times for \textsc{BronKerboschDelta}.

Algorithm VLM is usually faster than \textsc{BronKerboschDelta} for small values of \( \Delta \) below the \( \sim 5^3 \) threshold. Starting there, however, \textsc{BronKerboschDelta} clearly outperforms Algorithm VLM with running times smaller by at least one order of magnitude and up to three orders of magnitude (see Table 3). In terms of main memory, 385 MB is the maximum used by \textsc{BronKerboschDelta} over all solved instances, attained on infectious for \( \Delta = 0 \). On this instance, Algorithm VLM uses 494 MB and often more than one 1 GB.

Finally we mention that, increasing the time limit to six hours, \textsc{BronKerboschDelta} can solve all instances of karlsruhe with \( \Delta = 0 \) and \( \sim 5^1, 5^3, 5^5, 5^7, 5^9 \) wherein the last value involves enumerating 43 m maximal \( \Delta \)-cliques.
6 Conclusion

We studied the algorithmic complexity of enumerating $\Delta$-cliques in temporal graphs. We adapted the Bron-Kerbosch algorithm, including the procedure of pivoting to reduce the number of recursion calls, to the temporal setting and provided a theoretical analysis. In experiments on real-world data sets, we showed that our algorithm is notably faster than the first approach for enumerating all maximal $\Delta$-cliques in temporal graphs due to Viard et al. \cite{24, 25}. Our experimental results further reveal that pivoting can notably decrease the running time for large $\Delta$. Furthermore, we measured the $\Delta$-slice degeneracy for different $\Delta$-values and showed that it is reasonably small in many real-world data sets. The theoretical analysis (based on the $\Delta$-slice degeneracy parameter) of the running time still leaves room for improvement. In particular, we leave the impact of pivoting on the running time upper bound as an open question for future research.

References

[1] A. Barrat and J. Fournet. Contact patterns among high school students. *PLoS ONE*, 9(9):e107878, 2014.

[2] S. Boccaletti, G. Bianconi, R. Criado, C. I. Del Genio, J. Gómez-Gardeñes, M. Romance, I. Sendiña-Nadal, Z. Wang, and M. Zanin. The structure and dynamics of multilayer networks. *Physics Reports*, 544(1):1–122, 2014.

[3] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[4] D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in large sparse real-world graphs in near-optimal time. *ACM Journal of Experimental Algorithmics*, 18(3):3.1:1–3.1:21, 2013.

[5] T. Erlebach, M. Hoffmann, and F. Kammer. On temporal graph exploration. In *Automata, Languages, and Programming*, volume 9134 of LNCS, pages 444–455. Springer, 2015.
[6] V. Gemmetto, A. Barrat, and C. Cattuto. Mitigation of infectious disease at school: targeted class closure vs school closure. *BMC Infectious Diseases*, 14(1):1, 2014.

[7] R. Goerke. Email network of KIT informatics. [http://i11www.iti.uni-karlsruhe.de/en/projects/spp](http://i11www.iti.uni-karlsruhe.de/en/projects/spp), 2011.

[8] P. Holme and J. Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012.

[9] F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier. Isolation concepts for clique enumeration: Comparison and computational experiments. *Theoretical Computer Science*, 410(52):5384–5397, 2009.

[10] L. Isella, J. Stehlé, A. Barrat, C. Cattuto, J.-F. Pinton, and W. Van den Broeck. What’s in a crowd? analysis of face-to-face behavioral networks. *Journal of Theoretical Biology*, 271(1):166–180, 2011.

[11] H. Ito and K. Iwama. Enumeration of isolated cliques and pseudo-cliques. *ACM Transactions on Algorithms (TALG)*, 5(4):40, 2009.

[12] J. Kleinberg and É. Tardos. *Algorithm Design*. Pearson Education, 2006.

[13] C. Komusiewicz, F. Hüffner, H. Moser, and R. Niedermeier. Isolation concepts for efficiently enumerating dense subgraphs. *Theoretical Computer Science*, 410(38):3640–3654, 2009.

[14] M. Lahiri and T. Y. Berger-Wolf. Periodic subgraph mining in dynamic networks. *Knowledge and Information Systems*, 24(3):467–497, 2010.

[15] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 177–187. ACM, 2005.

[16] O. Michail. An introduction to temporal graphs: An algorithmic perspective. In *Algorithms, Probability, Networks, and Games*, volume 9295 of LNCS, pages 308–343. Springer, 2015.

[17] O. Michail and P. G. Spirakis. Traveling salesman problems in temporal graphs. In *Mathematical Foundations of Computer Science 2014*, volume 8635 of LNCS, pages 553–564. Springer, 2014.

[18] V. Nicosia, J. Tang, C. Mascolo, M. Musolesi, G. Russo, and V. Latora. Graph metrics for temporal networks. In *Temporal Networks*, Understanding Complex Systems, pages 15–40. Springer, 2013.

[19] T. Opsahl and P. Panzarasa. Clustering in weighted networks. *Social Networks*, 31(2):155–163, 2009.

[20] J. Stehlé, N. Voirin, A. Barrat, C. Cattuto, L. Isella, J.-F. Pinton, M. Quaggiotto, W. Van den Broeck, C. Régis, B. Lina, et al. High-resolution measurements of face-to-face contact patterns in a primary school. *PLoS ONE*, 6(8):e23176, 2011.

[21] E. Tomita, A. Tanaka, and H. Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363 (1):28–42, 2006.
[22] T. Uno and Y. Uno. Mining preserving structures in a graph sequence. *Theoretical Computer Science*, 2015.

[23] P. Vanhems, A. Barrat, C. Cattuto, J.-F. Pinton, N. Khanafé, C. Régis, B.-a. Kim, B. Comte, and N. Voirin. Estimating potential infection transmission routes in hospital wards using wearable proximity sensors. *PLoS ONE*, 8(9):e73970, 2013.

[24] J. Viard, M. Latapy, and C. Magnien. 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*, pages 1517–1522. ACM, 2015.

[25] J. Viard, M. Latapy, and C. Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016.