Revealing contact patterns among high-school students using maximal cliques in link streams

Tiphaine Viard, Matthieu Latapy, Clémence Magnien

▶ **To cite this version:**

Tiphaine Viard, Matthieu Latapy, Clémence Magnien. Revealing contact patterns among high-school students using maximal cliques in link streams. First International Workshop on Dynamics in Networks (DyNo15) in conjunction with the 2015 IEEE/ACM International Conference ASONAM, Aug 2015, Paris, France. pp.1517-1522, 10.1145/2808797.2809291 . hal-01208330v2

HAL Id: hal-01208330
https://hal.science/hal-01208330v2

Submitted on 18 Jul 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Revealing contact patterns among high-school students using maximal cliques in link streams

Tiphaine Viard*, Matthieu Latapy*, Clémence Magnien*
*Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606,
4 place Jussieu 75005 Paris

***Abstract*—Interaction traces between humans are usually rich in information concerning the patterns and habits of individuals. Such datasets have been recently made available, and more and more researchers address the new questions raised by this data. A link stream is a sequence of triplets (t, u, v) indicating that an interaction occurred between u and v at time t, and as such is a natural representation of these data. We generalize the classical notion of cliques in graphs to such link streams: for a given Δ, a Δ-clique is a set of nodes and a time interval such that all pairs of nodes in this set interact at least every Δ during this time interval. We proceed to compute the maximal Δ-cliques on a real-world dataset of contact among students, and show how it can bring new interpretation to patterns of contact.**

I. Introduction

Understanding the dynamics behind human interactions has a wide range of applications, from epidemic diffusion to mobility patterns modelling [13]. Recent advances in technology have allowed real-time tracking of interactions between groups of individuals, with RFID sensors, giving the scientific community a plethora of datasets to explore.

To study contact traces, it is natural to see such data as graphs where nodes are individuals and edges indicate that two individuals were in contact together. It is possible to obtain large graphs, enclosing much information on the structure of the interactions, and network science is a natural framework for studying them [9]. A common approach relies on sequences of graphs: for a given Δ, one considers the graph G_t induced by the interactions that occurred from time t to $t + \Delta$, then the graphs $G_{t+\Delta}$, $G_{t+2\Delta}$ and so on. Many variants exist, but the baseline remains that one splits time into (possibly overlapping) slices of given (but possibly evolving) duration Δ [12], [10]. Of course, an important problem with this approach is the (partial, or complete) destruction of the temporal dynamics. This is especially costly in contact traces, where interactions have been shown to be bursty, thus leading to graphs that are either densely connected or too small to analyze [4].

A link stream $L = (T, V, E)$ with V a set of nodes, $T = [\alpha, \omega]$ a time interval and $E \subseteq T \times V \times V$ models interactions over time: $l = (t, u, v)$ in E means that an interaction occurred between $u \in V$ and $v \in V$ at time $t \in T$. Link streams, also called temporal networks or time-varying graphs depending on the context, model many real-world data like contacts between individuals, email exchanges, or network traffic [3], [19], [8], [16].

For a given Δ, a Δ-clique C of L is a pair $C = (X, [b, e])$ with $X \subseteq V$ and $[b, e] \subseteq T$ such that for all $u \in X$, $v \in X$, and $\tau \in [b, e - \Delta]$, there is a link (t, u, v) in E with $t \in [\tau, \tau + \Delta]$.

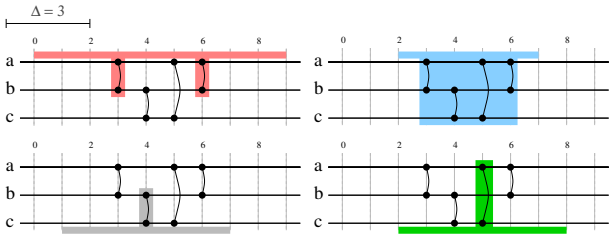

Fig. 1: Examples of Δ-cliques. We consider the link stream $L = ([0, 9], \{a, b, c\}, E)$ with $E = ((3, a, b), (4, b, c), (5, a, c), (6, a, b))$ and $\Delta = 3$. There are four maximal 3-cliques in L: $(\{a, b\}, [0, 9])$ (top left), $(\{a, b, c\}, [2, 7])$ (top right), $(\{b, c\}, [1, 7])$ (bottom left), and $(\{a, c\}, [2, 8])$ (bottom right). Notice that $(\{a, b, c\}, [1, 7])$ is not a Δ-clique since during time interval $[1, 4]$ of duration $\Delta = 3$ there is no interaction between a and c. Notice also that $(\{a, b\}, [1, 9])$, for instance, is not maximal: it is included in $(\{a, b\}, [0, 9])$.

More intuitively, all nodes in X interact at least once with each other at least every Δ from time b to time e. A clique C is maximal if it is included in no other clique, *i.e.* there exists no clique $C' = (X', [b', e']))$ such that $C' \neq C$, $X \subseteq X'$ and $[b, e] \subseteq [b', e']$. See Figure 1 for an example.

In real-world situations like the ones cited above, Δ-cliques are signatures of meetings, discussions, or distributed applications for instance. Moreover, just like cliques in a graph correspond to its subgraphs of density 1, Δ-cliques in a link stream correspond to its substreams of Δ-density 1, as defined in [16]. Therefore, Δ-cliques in link streams are natural generalizations of cliques in graphs.

In this paper, we propose a first algorithm for listing all maximal Δ-cliques of a given link stream, and illustrate the relevance of the concept and algorithm by computing maximal Δ-cliques of a real-world dataset of contacts between students. The results obtained shed new light on the nature of the interactions between students, and show new patterns of contact.

Notice that we consider here undirected links only: given a link stream $L = (T, V, E)$, we make no distinction between $(t, u, v) \in E$ and $(t, v, u) \in E$. Likewise, we suppose that there is no self-loop (t, v, v) in E, and no isolated node ($\forall v \in V,\ \exists (t, u, v) \in E$).

We finally define the first occurrence time of (u, v) after b as the smallest time $t \geq b$ such that $(t, u, v) \in L$, and we denote it by f_{buv}. Conversely we denote the last occurrence time of (u, v) before e by l_{euv}. We say that a link (t, u, v) is in $C = (X, [b, e])$ is $u \in X$, $v \in X$ and $t \in [b, e]$.

II. Algorithm

One may trivially enumerate all maximal cliques in a graph as follows. One maintains a set S of previously found cliques (maximal or not). Then for each C in S, one removes C from S and searches for nodes outside C connected to all nodes in C, thus obtaining new cliques (one for each such node) larger than C. If one finds no such node, then C is maximal and it is part of the output. Otherwise, one adds the newly found cliques to S. The set S is initialized with the trivial cliques containing only one node, and all maximal cliques have been found S is empty. [11] uses this framework to enumerate all the maximal cliques of a graph in lexicographic order.

Our algorithm for finding Δ-cliques in link stream $L = (T, V, E)$ (Algorithm 1) relies on the same scheme. We initialize the set S of found Δ-cliques with the trivial Δ-cliques $(\{a, b\}, [t, t])$ for all (t, a, b) in L (line 2). Then, until S is empty (*while* loop of lines 3 to 18), we pick an element $(X, [b, e])$ in S (line 4) and search for nodes v outside X such that $(X \cup \{v\}, [b, e])$ is a Δ-clique (lines 6 to 8). We also look for values $b' < b$ such that $(X, [b', e])$ is a Δ-clique (lines 9 to 12), and likewise values $e' > e$ such that $(X, [b, e'])$ is a Δ-clique (lines 13 to 16). If we find such a node, such a b' or such an e', then C is not maximal and we add to S the new cliques larger than C we just found (lines 8, 12 and 16). Otherwise, C is maximal and it is part of the output (line 18).

Algorithm 1 Maximal Δ-cliques of a link stream

input: a link stream $L = (T, V, E)$ and a duration Δ
output: the set of all maximal Δ-cliques in L

1: $S \leftarrow \emptyset$, $R \leftarrow \emptyset$
2: for $(t, u, v) \in E$: add $(\{u, v\}, [t, t])$ to S
3: **while** $S \neq \emptyset$ **do**
4: | take and remove $(X, [b, e])$ from S
5: | set isMax to True
6: | **for** v in $V \setminus X$ **do**
7: | | **if** $(X \cup \{v\}, [b, e])$ is a Δ-clique **then**
8: | | | add $(X \cup \{v\}, [b, e])$ to S and set isMax to False
9: | $f \leftarrow \max_{u,v \in X} f_{buv}$ ▷ *latest first occurrence time of a link in* $(X, [b, e])$
10: | **if** $b \neq f - \Delta$ **then**
11: | | let b' be $f - \Delta$
12: | | add $(X, [b', e])$ to S and set isMax to False
13: | $l \leftarrow \min_{u,v \in X} l_{euv}$ ▷ *earliest last occurrence time of a link in* $(X, [b, e])$
14: | **if** $e \neq l + \Delta$ **then**
15: | | let e' be $l + \Delta$
16: | | add $(X, [b, e'])$ to S and set isMax to False
17: | **if** isMax **then**
18: | | add $(X, [b, e])$ to R
19: **return** R

Let us explain the choice of b' (lines 10 to 12) in details, the choice of e' (lines 14 to 16) being symmetrical. Intuitively, we choose b' as small as possible, provided we do not miss any maximal Δ-clique. Therefore, for a given Δ-clique $(X, [b, e])$, we set b' to the smallest time such that $(X, [b', e])$ is a Δ-clique. This leads to the constraint of line 10: f is the latest of the first occurrence times of all links in the clique. If it is equal to $b + \Delta$ (line 10), then there is no $b' < b$ such that $(X, [b', e])$ is a Δ-clique. If it is different, then such a b' exists, and we set it to $f - \Delta$: by definition of f, all links in $X \times X$ appear at least once in $[b; f - \Delta]$. If there is a node $u \in V \setminus X$ such that $(X \cup \{u\}, [b, e])$ is a Δ-clique but $(X \cup \{u\}, [b', e])$ is not, $(X \cup \{u\}, [b, e])$ will be found separately; otherwise, all links in $\{u\} \times X$ exists in $[b; f - \Delta]$, by definition of a Δ-clique.

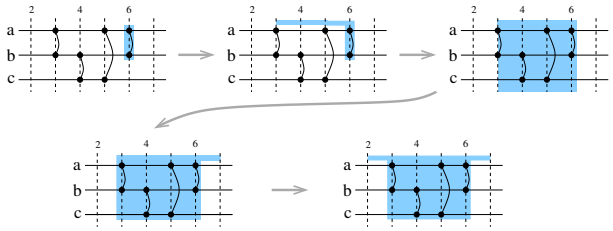

Fig. 2: A sequence of Δ-cliques built by our algorithm to find a maximal Δ-clique (bottom row) from an initial trivial Δ-clique (top-left) in the link stream of Figure 1 when $\Delta = 3$. From left to right and top to bottom: the algorithm starts with $(\{a, b\}, [6, 6])$, and finds $(\{a, b\}, [3, 6])$ thanks to lines 9 to 12 of the algorithm. It then finds $(\{a, b, c\}, [3, 6])$ thanks to lines 6 to 8. It finds $(\{a, b, c\}, [3, 7])$ from lines 13 to 16, and finally $(\{a, b, c\}, [2, 7])$ from lines 9 to 12.

We display in Figure 2 an example of a sequence of such operations from an initial trivial clique to a maximal clique in an illustrative link stream. The algorithm builds this way a set of Δ-cliques of L, which we call the *configuration space*; we display the configuration space for this simple example in Figure 3 together with the relations induced by the algorithm between these Δ-cliques.

To prove the validity of Algorithm 1, we must show that all the elements it outputs are cliques (1), that they are maximal (2), and that all maximal cliques are in its output (3). The full proof is available in [18].

(1) is proved by induction on the iterations of the *while* loop (lines 3 to 18). Initially, all elements of S are Δ-cliques. We assume that at the i-th iteration, S only contains Δ-cliques. The loop may add new elements to S at lines 8, 12 and 16. In all cases, the added element is built from an element $(X, [b; e])$ of S (line 4), which is a Δ-clique (induction hypothesis). From then on, it is easy to show that the elements added at lines 8, 12 and 16 are also Δ-cliques.

(2) is demonstrated by assuming that a non-maximal Δ-clique $(X, [b; e])$ is added to R (line 18), and by subsequently reaching a contradiction. Indeed, we show that from a Δ-clique C taken from S (line 4), the conditions at lines 7, 10 and 14 respectively check the existence of a node u in $V \setminus X$, a $b' < b$ or an $e' > e$ such that $(X \cup \{u\}, [b; e])$, $(X, [b'; e])$ or $(X, [b; e'])$ is a Δ-clique. If all these conditions are false, then C is maximal.

Finally, we prove the claim (3) by building a sequence

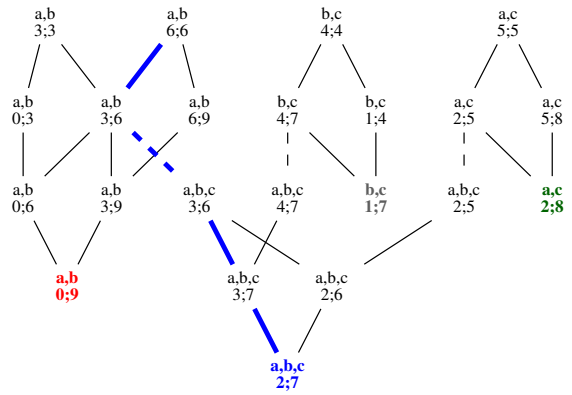

Fig. 3: The configuration space built by our algorithm from the link stream of Figures 1 and 2 when $\Delta = 3$. Each element is a Δ-clique and it is linked to the Δ-cliques the algorithm builds from it (links are implicitely directed from top to bottom). Plain links indicate Δ-cliques discovered by lines 9 to 12 or lines 13 to 16 of the algorithm, which change the time span of the clique. Dotted links indicate Δ-cliques discovered by lines 6 to 8, which change the set of nodes involved in the clique. The bold path is the one detailed in Figure 2. Colors correspond to the maximal Δ-cliques displayed in Figure 1.

$C_n, C_{n-1}, ..., C_0$, with C_n a maximal Δ-clique, such that for all Δ-cliques in this sequence, we have $C_{i-1} \rightarrow C_i$, meaning that if C_{i-1} was in S, C_i was added to S later in the execution of the algorithm by line 8, 12 or 16. Setting $C_0 = (\{u, v\}, [t; t])$ (line 2) proves the claim that all maximal Δ-cliques are in S at one point of the execution, and are trivially added to R at line 18.

Enumerating maximal cliques in graph $G = (V, E)$ is equivalent to enumerating maximal Δ-cliques in $L = ([0, 0], V, E')$ where $(0, u, v) \in E'$ if and only if $(u, v) \in E$. Therefore, enumerating Δ-cliques in a link stream is exponential (in particular the number of Δ-cliques may be exponential). To this regard, our algorithm is optimal: the number of elements of the configuration space built by the algorithm is bounded by the number of subsets of the set of links times the number of subsets of the set of link arrival times, and the number of operations performed for each of them is polynomial.

Still, several optimisations may speed up our algorithm (without changing its worst-case complexity), and we discuss some now.

First, Algorithm 1 may build and add to S the same Δ-clique many times. To avoid redundancy of computations, one may store the Δ-cliques seen so far. Then, a Δ-clique is added to S only the first time it is seen. This ensures that the number of runnings of the main loop is the number of links in the configuration space; in the naive version presented above, this number is the sum of the lengths of all paths from any initial trivial Δ-clique to the maximal ones it can reach.

Notice also that f and l, computed in lines 9 and 13, are necessarily in $[b, \min(e, b + \Delta)]$ and $[\max(b, e - \Delta), e]$, respectively. One may therefore focus the search on these intervals rather than $[b, e]$.

Going further, let us notice that if $V(C)$ is the set of nodes satisfying condition of line 7, then the set $V(C')$ of nodes satisfying this condition for the cliques C' added to S at lines 8, 12 and 16 is included in $V(C)$. One may therefore associate to each element of S a set of candidate nodes to be considered at line 6 in place of $V \setminus X$, thus drastically reducing the number of iterations of this loop.

Finally, notice that Algorithm 1 makes no assumption on the order in which elements of S are processed. This order corresponds to the way we explore the configuration space. In particular, if S is a first-in-first-out structure (like a queue), the algorithm performs a BFS of the configuration space; if it is a last-in-first-out structure (like a stack) then it performs a DFS. The execution time is the same in all cases. The size of S may vary, but the space complexity of the algorithm is dominated by the size of the set of already seen cliques just discussed, that does not change. Still, different exploration methods have different advantages and drawbacks. A BFS rapidly discovers all maximal cliques of small sizes and durations, which makes it suitable for computing the clique size distribution, or if one is interested in discovering as many maximal cliques as possible as rapidly as possible, even if they are all small ones. Conversely, a DFS first discovers large cliques and so it is appealing if one searches for non-trivial cliques.

III. Experiments

We implemented Algorithm 1 with all optimisations discussed above and provide the Python source code [17]. We illustrate here its practical relevance by computing maximal Δ-cliques of a link stream representing real-world contacts between individuals, captured with RFID sensors.

A. Dataset

This trace was collected at a french high school in 2012, see [7] for full details. It induces a link stream of 181 nodes and 45047 links, connecting 2220 distinct pairs of nodes over a period of 729500 seconds (approximately 8 days).

Each link (t, u, v) means that the sensor carried by individual u or v detected the sensor carried by the other individual at time t, which means in turn that these two individuals were close enough from each other at time t for the detection to happen. We call this a contact between individuals u and v. It is also worth to note that the dataset provides us with the classroom of each student. Also, the temporal resolution of the sensors is 20 seconds. Finally, the students only kept the sensors whilst inside the school, thus there are no contacts during the night or the weekend, as explained in [7].

In the remainder of this paper, we chose to list the maximal Δ-cliques for different values of Δ, namely 60 seconds, 900 seconds (15 minutes), 3600 seconds (1 hour). We handpicked these values after careful investigation regarding the rhythm of school day: indeed, on a typical day, courses usually last roughly two hours, with two 15 minutes breaks during the day, and a longer 1 hour lunch break. We also computed the Δ-cliques for $\Delta = 10800$ (3 hours) to have insight on a coarser temporal scale.

B. Computing Δ-cliques

For all the chosen Δs, a Δ-clique $C = (X, [b, e])$ means here that the individuals in set X were pairwise in contact at least once every 60, 900, 3600 or 10800 seconds from time b to e. These four values of Δ reveal different patterns among students at different temporal scales.

Our Python implementation succeeds in finding all the maximal Δ-cliques in this link stream in less than an hour on a standard computer. Although many discovered cliques are very small, we also find rather large and long cliques: more information concerning the Δ-cliques found is presented in table I.

Δ	# of Δ-cliques	Max $\|X\|$	Max $e - b$	Runtime (standard computer)
60	14664	5	6820	2 mn 30
900 (15 mns)	8214	7	17420	9 mn 15
3600 (1 hour)	7170	7	36340	18 mn 00
10800 (3 hours)	7416	7	59560	51 mn 40

TABLE I: Global information about the maximal Δ-cliques found.

To gain insight on the role played by the parameter Δ on the found maximal Δ-cliques, we compute the complementary cumulative distribution functions for the size of the Δ-cliques ($|X|$) and their duration ($e - b$). The result is visible in figure 4. As one might expect, the value of Δ positively correlates with larger and longer Δ-cliques.

Intuitively, regarding the size of the Δ-cliques (figure 4, left), a smaller Δ will group links inside a burst, but not outside of it, and as such the cliques tend to involve less nodes. This intuition is also backed up by the fact that the smaller the Δ, the more Δ-cliques are found, as we saw in table I. This is because for a larger value $\Delta' > \Delta$, small Δ-cliques tend to merge together into one larger Δ'-clique.

For every Δ, a sharp drop can be observed in the distribution of the durations (figure 4, right); this corresponds to the value $2 \cdot \Delta$. This behaviour is typical of Δ-cliques involving only two nodes ($|X| = 2$). When Δ is larger, the proportion of such Δ-cliques drops, as there are less Δ-cliques of size 2, which is confirmed in figure 4 (left). Conversely, the tails of the distributions get longer, as there are more larger and longer Δ-cliques.

Let us define the surrounding stream of a Δ-clique. For a maximal Δ-clique $(X, [b, e])$ and a stream $L = (T, V, E)$, the surrounding stream is the substream $L' = (T, V, E')$, with $E' = \{(t, u, v) : t \in [b, e], u \in V, v \in V\}$. Intuitively, the surrounding stream contains all the links (t, u, v) having $t \in [b, e]$. We show in figure 5 two typical Δ-cliques found in the contact trace, along with their surrounding stream. The tool for drawing link streams is available online [15]. It gives us an intuitive lookup onto the behaviours of the nodes involved in the Δ-cliques we found.

The 60-clique exhibited in figure 5 (left) is a clique between 5 students of different classrooms, the largest size found for this Δ. The duration of this clique is very small (~3 minutes), and converting the timestamps to dates show these students were in contact during the lunch break (from 12:47 to 12:50). It is likely to be a group of friends that gathered before the beginning of the afternoon courses, or a meeting between the classroom's representatives.

The 60-clique shown in figure 5 (right) is the longest one found by our algorithm: it lasts for 1080 seconds (18 minutes), and, surprisingly, involves two students from different classrooms. Considering the short range of the sensors, it is unlikely that this contact is due to students being close to each other, but in different rooms. The timestamp of the 60-clique, around 10am, likely points to an in-between courses discussion between two students.

C. Comparison with cliques in the aggregated graph

In this section, we further investigate the descriptive role of Δ-cliques as compared to cliques in the aggregated graph. For a link stream $L = (T, V, E)$, as defined in section I, we define the aggregated graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = V$ and $\mathcal{E} = \{(u, v) : \exists (t, u, v) \in E\}$. Intuitively, this is the graph obtained by putting a link between two nodes u and v if and only if there is at least one contact between u and v in the link stream.

We then proceed to compute the maximal cliques of $\mathcal{G}$. This yields 1742 cliques, the largest one involving 14 nodes. Many of these cliques have overlapping nodes.

Remember that in the aggregated graph, two nodes are linked if there is at least one contact between these nodes, regardless of the time and frequency of their interaction. As such, cliques in $\mathcal{G}$ are typically linking students in the same classroom. Most of the cliques (~70%) involve students of the same class.

Considering Δ-cliques instead sheds light on different patterns. Of course, the very definition of a Δ-clique involves a clique in the corresponding aggregated graph. As such, it is possible to find the students' classrooms solely by looking at the Δ-cliques. However, other patterns are also highlighted in Δ-cliques. For instance, we found a 60-clique involving 4 students of different classrooms during roughly 5 minutes, which is likely to be the signature of coffee breaks.

Considering the 3600-cliques, we notice that most of them last a few hours. This is typically courses; our hypothesis is that no more than 7 nodes are found interacting together because of the sensor's range (roughly 1 meter), meaning that students sharing the same classroom will not all be pairwise in contact. The daily lunch break provides an explanation for the fact that few 3600-cliques last longer than 3 hours. Indeed, students might eat outside (thus giving back the sensors), or eat with friends from different classes (thus ending the previous 3600-cliques).

The 900-cliques have an intermediary interpretation, allowing us to clearly see the distinction between time in and out of the classroom. Some cliques (usually involving students of the same class) last during a few hours (up to 4 hours), whereas others are much shorter (around 25 minutes), corresponding to contacts in between classrooms.

The rhythm of the school day explains the large gaps between the maximum durations found for different Δs. Effectively, the students being in contact for 15 minutes long

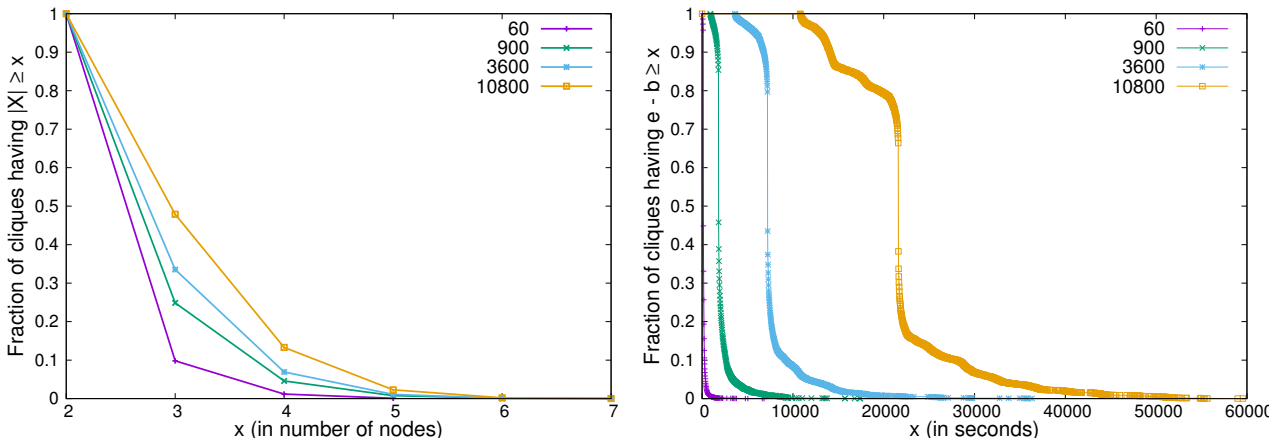

Fig. 4: **Left:** the complementary cumulative distribution function of clique sizes for different values of Δ. **Right:** the complementary cumulative distribution function of clique durations for different values of Δ. The sharp drop at $2 \cdot \Delta$ is due to Δ-cliques involving only 2 nodes.

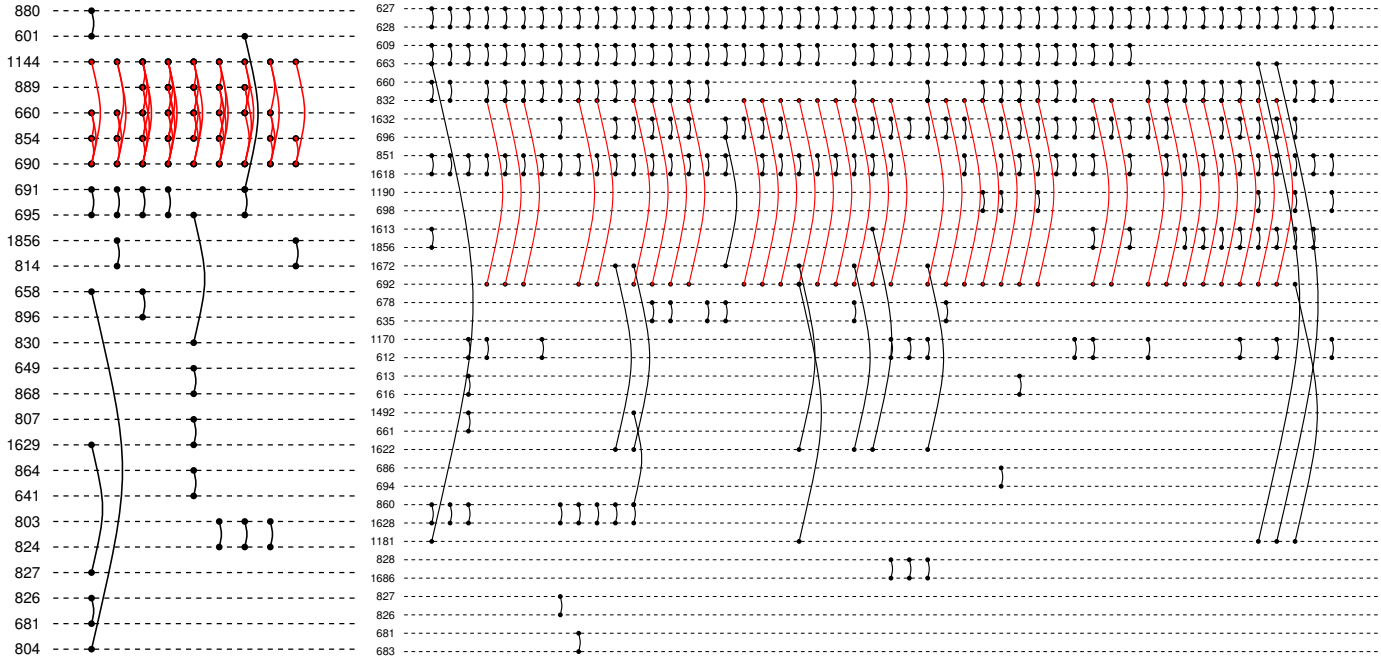

Fig. 5: Two 60-cliques (in red) in their surrounding stream. **Left:** The 60-clique $(\{1144, 889, 660, 854, 690\}, 1353325660, 1353325820)$. For scale, the time between the two first occurrences of the link $(1144, 690)$ corresponds to 20 seconds. **Right:** The 60-clique $(\{832, 692\}, 1353920500, 1353921480)$. For scale, the time between the two first occurrences of the link $(832, 692)$ corresponds to 20 seconds.

are either during a 15 minute break or in the middle of a 2 hour course, and the condition of being in contact at least once every 15 minutes is easily fulfilled. Such interactions are likely to stop with lunch breaks or the end of the school day, which explains that the maximum duration found for 900 is 4 hours. The maximal duration found for $\Delta = 3600$ (1 hour) is likely to be due to students having a short lunch breaking, thus perpetuating their 3600-clique through the school day.

These long Δ-cliques were not found for $\Delta = 60$; we attribute this both to the short spatial resolution of the sensors (~1 meter) and the short temporal scale (1 minute): a student being called at the whiteboard, for instance, would easily break the conditions to make a 60-clique.

A larger Δ (for example 3 hours) exhibits other kinds of behaviours, as it allows Δ-cliques to go through the 1 hour long lunch break. Though obtaining a full result with such a Δ is out of reach with the current state of our algorithm, we managed to obtain a good estimation of the largest cliques for $\Delta = 10800$. Indeed, as stated in section II, going through the configuration space using a depth-first search allows us to obtain maximal cliques faster. Plus, since long or large Δ-cliques statistically involve more links in the corresponding

link stream, picking links randomly from the dataset results in a higher likelihood of finding large or long Δ-clique quickly. Indeed, this intuition is right, as $\Delta = 10800$ raise a much longer Δ-clique, lasting 59560 seconds (approximately 16 hours) within a few minutes of computation. This Δ is large enough to embrace the whole school day (approximately 10 hours, from 9am to 6pm), but not enough to go through the night.

It is interesting to note that looking at graph snapshots over time, i.e. a sequence of graphs $\{G_t\}$ induced by the interactions that occurred from time t to $t + \Delta$, then the graphs $G_{t+\Delta}$, $G_{t+2\Delta}$ and so on, will yield different results. Choosing where a snapshot starts and ends is arbitrary, and will split cliques between two snaphots, making those undetected. Instead, Δ-cliques consider time as a continuous value and will raise all such structures.

As far as we know, the detection of such patterns that involve both structure and time with one simple dedicated notion is novel, and is not raised by classical clique detection. Focusing solely on the temporal aspects (for instance, looking at the intercontact distribution, as done in [7]), does not lead to identifying those patterns between students of the same or of different classrooms.

IV. Conclusion

We introduce the notion of Δ-clique in link streams as a generalization of cliques in graphs. We provide a first algorithm to list all the maximal Δ-cliques. Running it on contact among high-school students revealed new patterns of contact at different temporal scales, opening the way for a better understanding of human contact patterns.

Clearly, our algorithm may be further improved. Trying to adapt the Bron-Kerbosch algorithm [2] and some of its variants [14], [5], [1], [6], the most widely used algorithms for computing cliques in graphs, is particularly appealing. Indeed, the configuration spaces built by these algorithms are trees, and they are much smaller than our configuration spaces. This is achieved by maintaining a set of candidate nodes that may be added to previously discovered cliques, which does not directly translate to our situation because of time in link streams. Still, we believe that progress is possible in this direction.

Another perspective is to extract the maximal Δ'-cliques, with $\Delta' < \Delta$ and given the knowledge of the Δ-cliques (both maximal and non-maximal). This is a significant improvement, as one only needs to do a costly computation once (for a sufficiently large Δ). This is possible because for a $\Delta' < \Delta$, all the maximal Δ'-cliques will be contained at one point in the configuration space created for computing the Δ-cliques.

We also consider the case of links with duration as a promising perspective: each link (b, e, u, v) means that u and v are in continuous contact from time b to e. In this case there is no need for a Δ anymore, as density in this context is nothing but the probability that two randomly chosen nodes are linked together at a randomly chosen time. The definition of cliques in link streams with durations follows directly, and our algorithm may be extended to compute the maximal such cliques.

Acknowledgments.

This work is supported in part by the french *Direction Générale de l'Armement* (DGA), by the CODDDE ANR-13-CORD-0017-01 grant from the *Agence Nationale de la Recherche*, and by grant O18062-44430 of the French program *PIA – Usages, services et contenus innovants*.

References

[1] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65:21–46, 1996.

[2] Coen Bron and J Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9), 1973.

[3] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *CoRR*, abs/1012.0009, 2010.

[4] Ciro Cattuto, Wouter Van den Broeck, Alain Barrat, Vittoria Colizza, Jean-François Pinton, and Alessandro Vespignani. Dynamics of person-to-person interactions from distributed rfid sensor networks. *PLoS ONE*, 5(7):e11596, 07 2010.

[5] Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal of Computing*, 14(1):210–223, 1985.

[6] David Eppstein and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. *Experimental Algorithms*, pages 364–375, 2011.

[7] Julie Fournet and Alain Barrat. Contact patterns among high school students. *PLoS ONE*, 9:e107878, 2014.

[8] Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519:97–125, 2012.

[9] Theus Hossmann, Thrasyvoulos Spyropoulos, and Franck Legendre. A complex network analysis of human mobility. In *NETSCICOM 2011, 3rd IEEE International Workshop on Network Science for Communication Networks, in conjuction with IEEE INFOCOM 2011, April 15, 2011, Shanghai, China*, Shanghai, CHINE, 04 2011.

[10] Theus Hossmann, Thrasyvoulos Spyropoulos, and Franck Legendre. Putting contacts into context: Mobility modeling beyond inter-contact times. In *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '11, pages 18:1–18:11, New York, NY, USA, 2011. ACM.

[11] David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 3 1988.

[12] Anna-Kaisa Pietiläinen and Christophe Diot. Dissemination in opportunistic social networks: The role of temporal communities. In *Proceedings of the Thirteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '12, pages 165–174, New York, NY, USA, 2012. ACM.

[13] Michele Starnini, Andrea Baronchelli, and Romualdo Pastor-Satorras. Modeling human dynamics of face-to-face interaction networks. *Phys. Rev. Lett.*, 110:168701, Apr 2013.

[14] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363:28–42, 2006.

[15] Tiphaine Viard. Source code in python for drawing link streams: `https://github.com/TiphaineV/LinkStreamViz`, 2014.

[16] Tiphaine Viard and Matthieu Latapy. Identifying roles in an IP network with temporal and structural density. In *Computer Communications Workshops (INFOCOM WKSHPS)*, pages 801–806, 2014.

[17] Tiphaine Viard and Matthieu Latapy. Source code in python for computing cliques in link streams: `https://github.com/TiphaineV/delta-cliques`, 2014.

[18] Tiphaine Viard, Matthieu Latapy, and Clémence Magnien. Computing maximal cliques in link streams. *submitted*, February 2015.

[19] Klaus Wehmuth, Artur Ziviani, and Eric Fleury. A Unifying Model for Representing Time-Varying Graphs. Research Report RR-8466, ENS Lyon, 2014.