Calcul de cliques maximales dans les flots de liens

Tiphaine Viard[1], Matthieu Latapy[1], Clémence Magnien[1]

[1] *Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France*
CNRS, UMR 7606, LIP6, F-75005, Paris, France

Un flot de liens est une séquence de triplets (t,u,v), signifiant que u et v ont interagi au temps t. Nous généralisons la notion de cliques à ces flots de liens : pour un delta donné, une delta-clique est un ensemble de noeuds et un intervalle de temps, tels que toutes les paires de noeuds dans cet ensemble interagissent au moins tous les delta sur cet intervalle. Nous proposons un premier algorithme permettant d'énumérer les delta-cliques dans un flot de liens.

Keywords: link streams, temporal networks, time-varying graphs, cliques, graphs, algorithms

1 Introduction

A link stream $L = (T, V, E)$ with $T = [\alpha, \omega]$ 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, 10, 6, 8].

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 $t \in [\tau, \tau + \Delta]$.

More intuitively, all nodes in X interact at least once with each other at least every Δ from time b to time e. Clique C is maximal if it is included in no other clique, *i.e.* there exists no clique $C' = (X', [b', e']))$ such that $X' \subset X$ or $[b', e'] \subset [b, e]$. 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 [8]. 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. Before entering in the core of the presentation, notice that we consider here undirected links only. Likewise, we suppose that there are no loops, and no isolated nodes.

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 $t \in [b, e]$.

2 Algorithm

One may trivially enumerate all maximal cliques in a graph as follows. One maintains a set S of previously found cliques that may be 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 when S is empty.
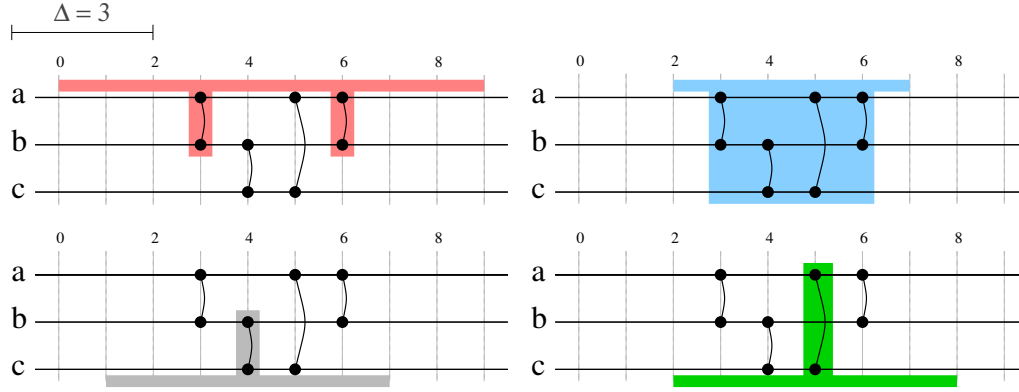
Fig. 1: Examples of Δ-cliques. We consider the link stream $L = ([0,9], \{a,b,c\}, E)$ with $E = ((3,a,b),(4,b,c),(5,a,c),(6,a,b))$ and $\Delta = 3$. There are four maximal 3-cliques in L: $(\{a,b\},[0,9])$ (top left), $(\{a,b,c\},[2,7])$ (top right), $(\{b,c\},[1,7])$ (bottom left), and $(\{a,c\},[2,8])$ (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 $(\{a,b\},[0,9])$.

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:          **if** $\exists (t,u,v) \in E, f - \Delta \leq t < b$ and $\{u,v\} \cap X \neq \emptyset$ **then**
12:              let b' be the maximal such t
13:          **else**
14:              let b' be $f - \Delta$
15:          add $(X,[b',e])$ to S and set isMax to False
16:      $l \leftarrow \min_{u,v \in X} l_{euv}$ ▷ *earliest last occurrence time of a link in $(X,[b,e])$*
17:      **if** $e \neq l + \Delta$ **then**
18:          **if** $\exists (t,u,v) \in E, e < t \leq l + \Delta$ and $\{u,v\} \cap X \neq \emptyset$ **then**
19:              let e' be the minimal such t
20:          **else**
21:              let e' be $l + \Delta$
22:          add $(X,[b,e'])$ to S and set isMax to False
23:      **if** isMax **then**
24:          add $(X,[b,e])$ to R
25: **return** R

Our algorithm for finding Δ-cliques in a 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 24), 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
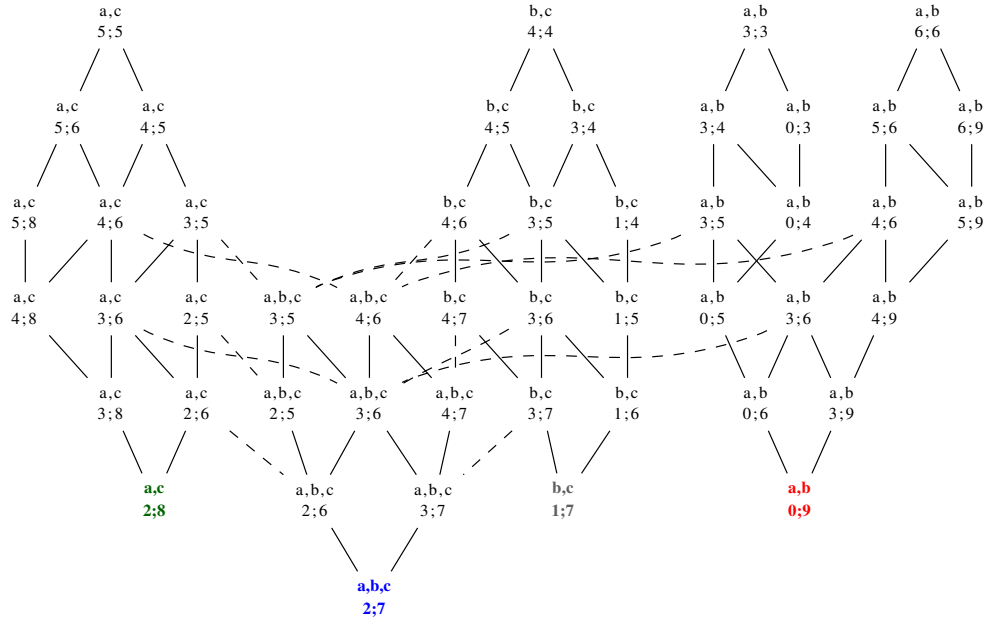
Fig. 2: The configuration space built by our algorithm from the link stream of Figure 1 when $\Delta = 3$. Each element is a Δ-clique and it is linked to the Δ-cliques the algorithm builds from it (links are implicitly directed from top to bottom). Plain links indicate Δ-cliques discovered from lines 9 to 15 or lines 16 to 22 of the algorithm, which change the time span of the clique. Dashed links indicate Δ-cliques discovered from lines 6 to 8, which change the set of nodes involved in the clique. Colors correspond to the maximal Δ-cliques displayed in Figure 1.

$b' < b$ such that $(X, [b', e])$ is a Δ-clique (lines 9 to 15), and likewise values $e' > e$ such that $(X, [b, e'])$ is a Δ-clique (lines 16 to 22). 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, 15 and 22). Otherwise, C is maximal and it is part of the output (line 24).

As time is a continuous quantity, finding appropriate values for b' and e' above is non-trivial. 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 latest time a link involving a node in X occurred before b: this link may make it possible to add a node to the Δ-clique. We also have to ensure that the obtained object remains a Δ-clique after the transformation. This leads to the two constraints of lines 10 and 17: f is the latest of the first occurrences 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 choose it in lines 11–15: we search for the latest link before b that involves a node in X; if it occurs after $f - \Delta$ then we choose b' as the time of this link; otherwise we set b' to $f - \Delta$, which is the smallest possible value such that $(X, [b', e])$ is a clique. The scheme for choosing e' is similar (lines 17 to 22).

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 2 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 [9].

(1) is proved by induction on the iterations of the *while* loop (lines 3 to 24). 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, 15 and 22. 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, 15 and 22 are also Δ-cliques.

(2) is demonstrated by assuming that a non-maximal Δ-clique $(X, [b; e])$ is added to R (line 24), 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 17 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 $C_n, C_{n-1}, ..., C_0$, such that for all Δ-cliques in this sequence, we have $C_{i-1} \to C_i$, meaning that if C_{i-1} was in S, C_i was added to S earlier in the execution of the algorithm by line 8, 15 or 22. 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 24.

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.

3 Conclusion

We introduced the notion of Δ-cliques in link streams, and proposed a first algorithm to compute the maximal such cliques. Clearly, our algorithm may be improved further. Trying to adapt the Bron-Kerbosch algorithm [2] and some of its variants [7, 4, 1, 5], which are 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 own configuration spaces.

References

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

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

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

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

[7] 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.

[8] 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.

[9] Tiphaine Viard, Matthieu Latapy, and Clémence Magnien. Computing maximal cliques in link streams. *ArXiv e-prints*, February 2015.

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