Theoretical Computer Science 609 (2016) 245–252

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Computing maximal cliques in link streams

Jordan Viard *, Matthieu Latapy, Clémence Magnien

Sorbonne Universités, UPMC Univ. Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu, 75005 Paris, France

ARTICLE INFO

Article history:
Received 10 February 2015
Received in revised form 23 September 2015
Accepted 28 September 2015
Available online 3 October 2015
Communicated by P. Widmayer

Keywords:
Link streams
Temporal networks
Time-varying graphs
Cliques
Graphs
Algorithms

ABSTRACT

A link stream is a collection of triplets (t, u, v) indicating that an interaction occurred between u and v at time t. 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 once during each sub-interval of duration Δ. We propose an algorithm to enumerate all maximal (in terms of nodes or time interval) cliques of a link stream, and illustrate its practical relevance to a real-world contact trace.

1. Introduction

In a graph $G = (V, E)$ with $E \subseteq V \times V$, a clique $C \subseteq V$ is a set of nodes such that $C \times C \subseteq E$. In addition, C is maximal if it is included in no other clique. In other words, a maximal clique is a set of nodes such that all possible links exist between them, and there is no other node linked to all of them. Enumerating maximal cliques of a graph is one of the most fundamental problems in computer science, and it has many applications [9,10].

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 [2,6,12,14].

For a given duration 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 $|X| \geq 2$, and for all $\{u, v\} \subseteq X$ and $\tau \in [b, \max(e - \Delta, b)]$ there is a link (t, u, v) in E with $t \in [\tau, \min(\tau + \Delta, e)]$. Notice that Δ-cliques necessarily have at least two nodes.

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

* Corresponding author.
E-mail address: jordan.viard@lip6.fr (J. Viard).

http://dx.doi.org/10.1016/j.tcs.2015.09.030
0304-3975/

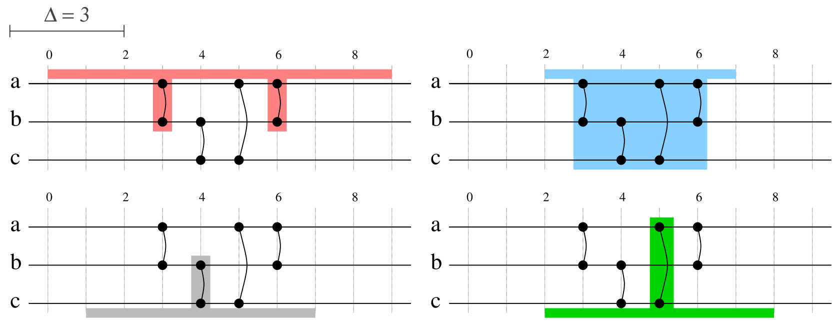

Fig. 1. Examples of Δ-cliques. We consider the link stream $L = ([0,9], \{a,b,c\}, E)$ with $E = \{(3,a,b), (4,b,c), (5,a,c), (6,a,b)\}$ and $\Delta = 3$. There are four maximal 3-cliques in L: $(\{a,b\},[0,9])$ (top-left), $(\{a,b,c\},[2,7])$ (top-right), $(\{b,c\},[1,7])$ (bottom-left), and $(\{a,c\},[2,8])$ (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 $(\{a,b\},[0,9])$.

stream correspond to its substreams of Δ-density 1, as defined in [12]. Therefore, Δ-cliques in link streams are natural generalizations of cliques in graphs.

In this paper, we propose the first algorithm for listing all maximal Δ-cliques of a given link stream. We illustrate the relevance of the concept and algorithm by computing maximal Δ-cliques of a real-world dataset.

Before entering in the core of the presentation, 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 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 $(t,u,v) \in E$, 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 $C = (X,[b,e])$ if $u \in X$, $v \in X$ and $t \in [b,e]$.

2. Algorithm

One may trivially enumerate all maximal cliques in a graph as follows. One maintains a set M of previously found cliques (maximal or not), as well as a set S of candidate cliques. Then for each clique C in S, one removes C from S and searches for nodes outside C connected to all nodes in clique C, thus obtaining new cliques (one for each such node) larger than C. If one finds no such node, then clique C is maximal and it is part of the output. Otherwise, if the newly found cliques have not already been found (i.e., they do not belong to M), then one adds them to S and M. The set S is initialized with the trivial cliques containing only one node, and all maximal cliques have been found when S is empty. The set M is used for memorization, and ensures that one does not examine the same clique more than once. In [7] the authors use this framework to enumerate all 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 candidate Δ-cliques and the set M of all found Δ-cliques with the trivial Δ-cliques $(\{a,b\},[t,t])$ for all (t,a,b) in E (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 10). We also look for a value $b' < b$ such that $(X,[b',e])$ is a Δ-clique (Lines 11 to 16), and likewise a value $e' > e$ such that $(X,[b,e'])$ is a Δ-clique (Lines 17 to 22). If we find such a node, such a b' or such an e', then Δ-clique C is not maximal and we add to S and M the new Δ-cliques larger than C we just found (Lines 10, 16 and 22), on the condition that they had not already been seen (i.e., they do not belong to M). Otherwise, C is maximal and is part of the output (Line 24).

Let us explain the choice of b' (Lines 11 to 16) in details, the choice of e' (Lines 19 to 22) being symmetrical. For a given Δ-clique $(X,[b,e])$, we set b' to $f - \Delta$, which is the smallest time such that we are sure that $(X,[b',e])$ is a Δ-clique without inspecting any link outside of $(X,[b,e])$. Indeed, all links in $X \times X$ appear at least once in the interval $[f-\Delta, f]$: f is the latest of the first occurrence times of all links in this Δ-clique, and so all links appear at least once in $[b,f] \subseteq [f-\Delta, f]$. If $b' \neq b$, then the Δ-clique $(X,[b',e])$ is added to S (Line 13).

We display in Fig. 2 an example of a sequence of such operations from an initial trivial Δ-clique to a maximal Δ-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 Fig. 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 Δ-cliques, that they are maximal, and that all maximal Δ-cliques are in its output.

Lemma 1. *In Algorithm 1, all elements of S are Δ-cliques of L.*

Proof. We prove the claim by induction on the iterations of the *while* loop (Lines 3 to 24).