Faster maximal clique enumeration in large real-world link streams

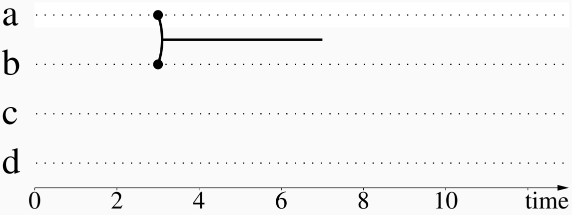
Alexis Baudin

January 30, 2024

IRIF – Graph seminar

Table of contents

1 - Maximal clique enumeration in link streams



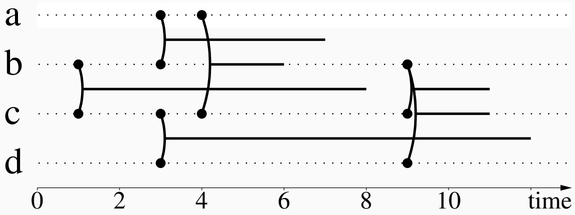
Link stream

- Vertices : a, b, c et d
- Time period : [0,12]
- Interaction : temporal links
  - a, b linked over $[3, 7]$

Advantages

- deals directly with the stream of interactions
- no arbitrary choice of time scale
- time is continuous

1 - Maximal clique enumeration in link streams
> Link stream: model temporal interactions



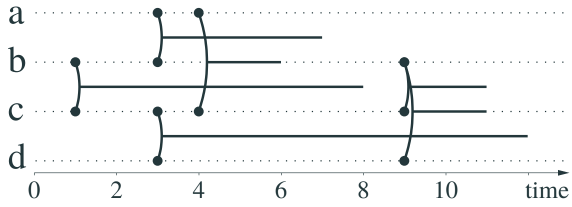
Link stream

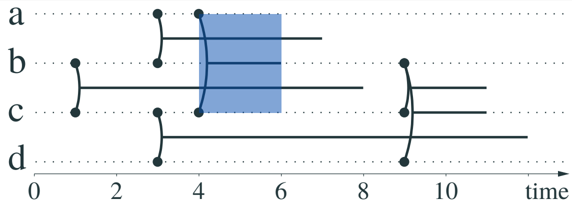
- Vertices : a, b, c et d
- Time period : [0,12]
- Interaction : temporal links
  - a, b linked over $[3, 7]$
  - ...

Advantages

- deals directly with the stream of interactions
- no arbitrary choice of time scale
- time is continuous

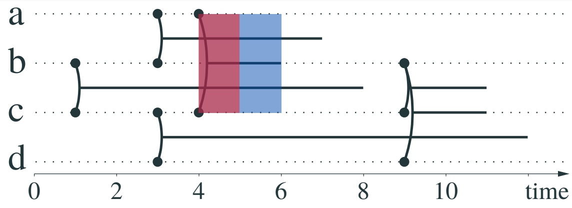
1 - Maximal clique enumeration in link streams
> Maximal cliques in link streams





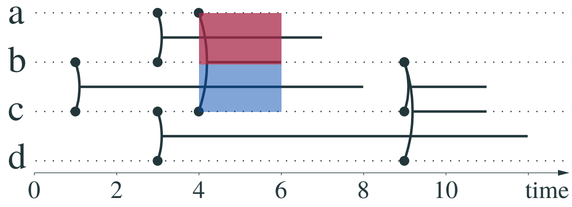
$(\{a, b, c\}, [4, 6])$ **is a clique**

1 - Maximal clique enumeration in link streams
> Maximal cliques in link streams



$(\{a, b, c\}, [4, 6])$ **is a clique**

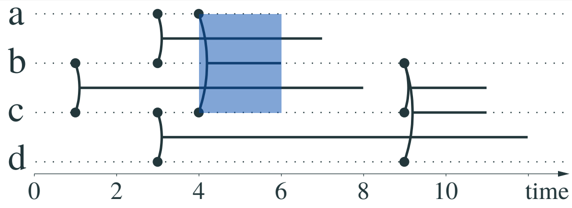
$\rightarrow$ $(\{a, b, c\}, \underline{[4, 5]})$ is not time-maximal



$(\{a, b, c\}, [4, 6])$ **is a clique**

→ $(\{a, b, c\}, \underline{[4, 5]})$ is not time-maximal

→ $(\underline{\{a, b\}}, [4, 6])$ is not vertex-maximal

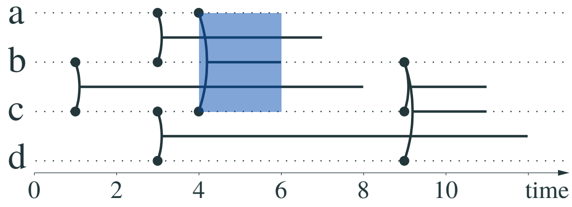


$(\{a, b, c\}, [4, 6])$ **is a clique**

→ $(\{a, b, c\}, \underline{[4, 5]})$ is not time-maximal

→ $(\underline{\{a, b\}}, [4, 6])$ is not vertex-maximal

$\boxed{\textbf{maximal clique} = \textit{time-maximal} \text{ and } \textit{vertex-maximal}}$

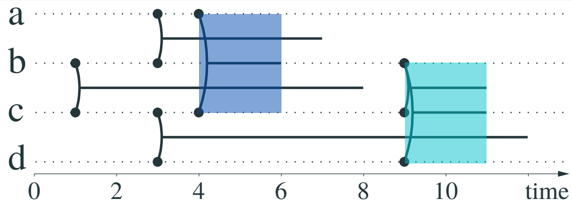


$(\{a, b, c\}, [4, 6])$ **is a clique (maximal).**

$\rightarrow$ $(\{a, b, c\}, \underline{[4, 5]})$ is not time-maximal

$\rightarrow$ $(\underline{\{a, b\}}, [4, 6])$ is not vertex-maximal

maximal clique = *time-maximal* and *vertex-maximal*

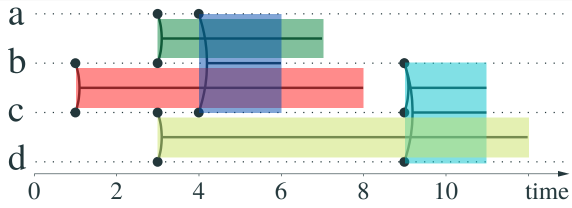


$(\{a, b, c\}, [4, 6])$ **is a clique (maximal).**

→ $(\{a, b, c\}, \underline{[4, 5]})$ is not time-maximal

→ $(\underline{\{a, b\}}, [4, 6])$ is not vertex-maximal

maximal clique = *time-maximal* and *vertex-maximal*



$(\{a, b, c\}, [4, 6])$ **is a clique (maximal).**

→ $(\{a, b, c\}, \underline{[4, 5]})$ is not time-maximal

→ $(\underline{\{a, b\}}, [4, 6])$ is not vertex-maximal

maximal clique = *time-maximal* and *vertex-maximal*

How to efficiently enumerate maximal cliques in massive real-world link streams?

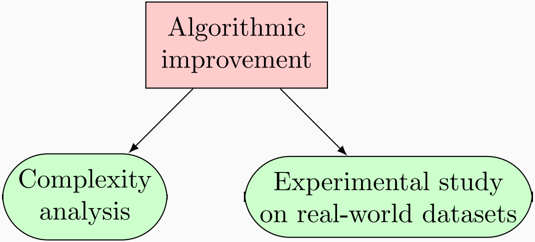
Input: A link stream.

Output: List of all its maximal cliques.

How to efficiently enumerate maximal cliques in massive real-world link streams?

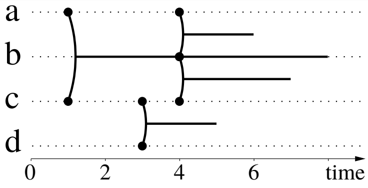
Input: A link stream.

Output: List of all its maximal cliques.



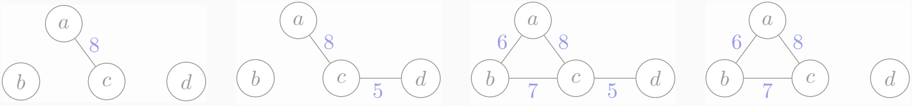
2 - New algorithm

List of temporal links sorted chronologically



↓

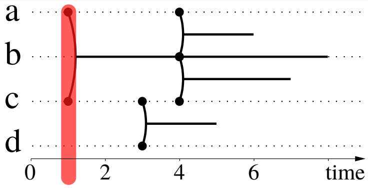
Evolving instantaneous graph G_t, with end dates



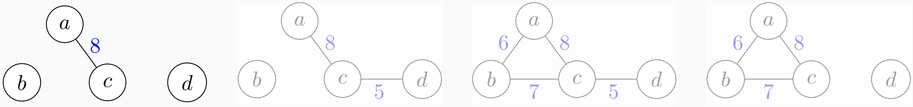
$G_1 : t = 1$ $G_3 : t = 3$ $G_4 : t = 4$ $G_5 : t = 5$

2 - New algorithm
> Input datastructure

List of temporal links sorted chronologically



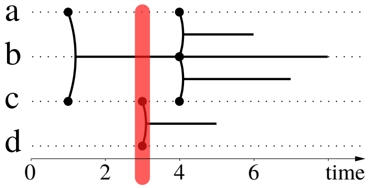
Evolving instantaneous graph G_t, with end dates



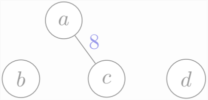
$G_1 : t = 1$ $G_3 : t = 3$ $G_4 : t = 4$ $G_5 : t = 5$

2 - New algorithm
> Input datastructure

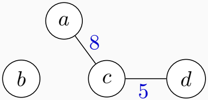
List of temporal links sorted chronologically



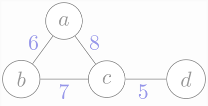
Evolving instantaneous graph G_t, with end dates



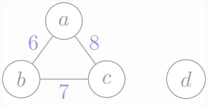
$G_1 : t = 1$



$G_3 : t = 3$

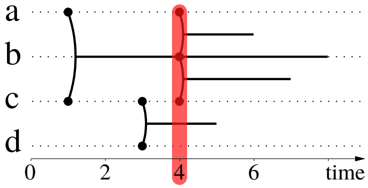


$G_4 : t = 4$

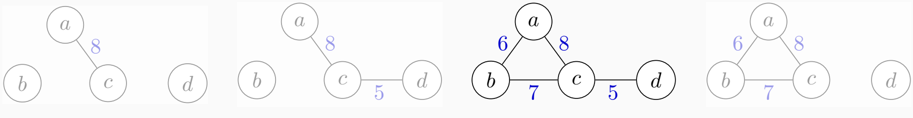


$G_5 : t = 5$

List of temporal links sorted chronologically

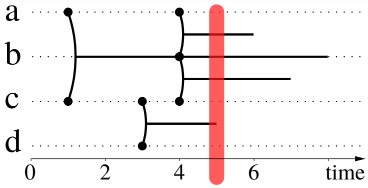


Evolving instantaneous graph G_t, with end dates



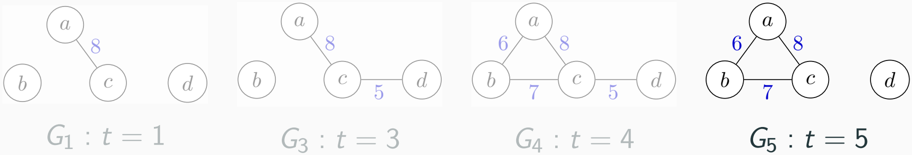
$G_1 : t = 1$ $G_3 : t = 3$ $G_4 : t = 4$ $G_5 : t = 5$

List of temporal links sorted chronologically

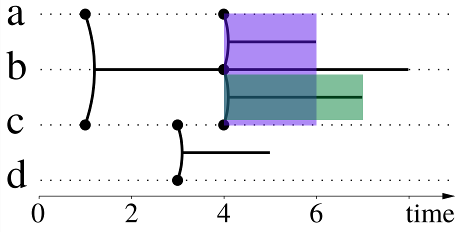


↓

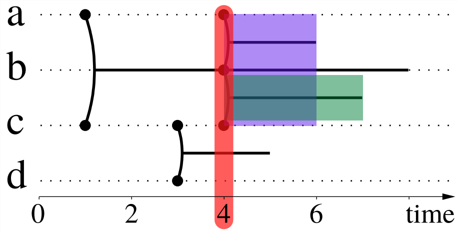
Evolving instantaneous graph G_t, with end dates



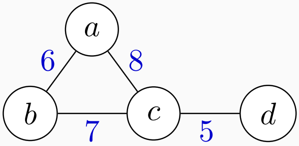
2 - New algorithm
> Maximal cliques that begin at each time t



2 - New algorithm
> Maximal cliques that begin at each time t

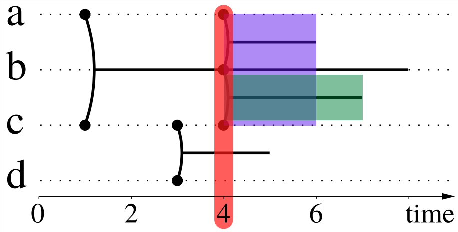


Instantaneous graph G_4

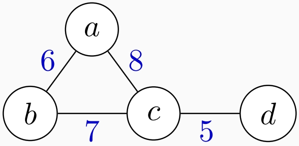


2 - New algorithm

> Maximal cliques that begin at each time t



Instantaneous graph G_4



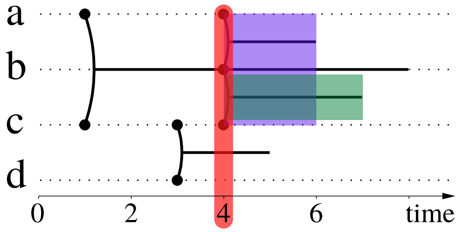
Cliques of G_4 that contain a new edge $\Longleftrightarrow$ Time-maximal cliques that begin at $t = 4$

Graph algorithm

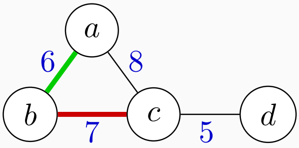
Instantaneous graph G_4

Filter

Maximal cliques that begin at $t = 4$



Instantaneous graph G_4

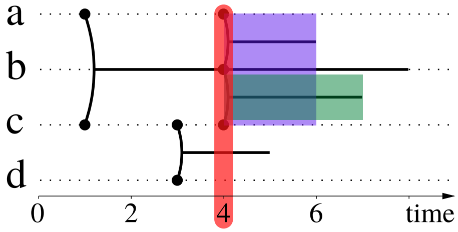


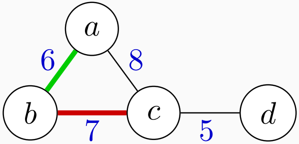
$(\{b, c\},\quad), (\{a, b, c\},\quad), (\{a, b\},\quad)$

$\rightarrow$ cliques of G_4 containing a new edge

2 - New algorithm

$>$ Maximal cliques that begin at each time t

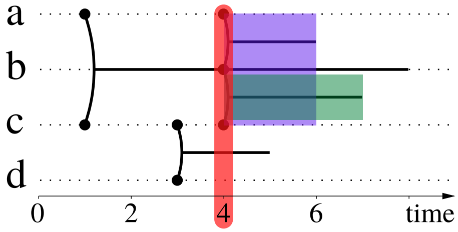




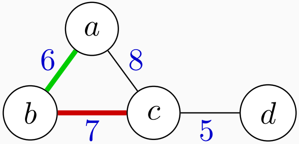
({b, c}, [4,), ({a, b, c}, [4,), ({a, b}, [4,)

$\rightarrow$ cliques of G_4 containing a new edge

$\rightarrow$ starting time: $t = 4$



Instantaneous graph G_4



({b, c}, [4, 7]), ({a, b, c}, [4, 6]), ({a, b}, [4, 6])

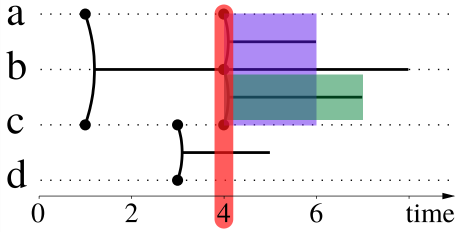
→ cliques of G_4 containing a new edge

→ starting time: $t = 4$

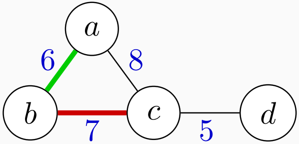
→ end time: min of edge end times

2 - New algorithm

$>$ Maximal cliques that begin at each time t

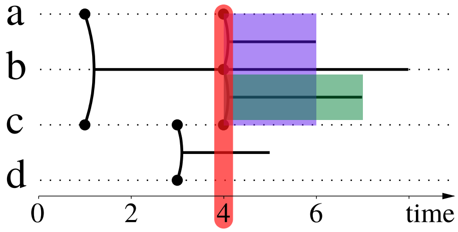


Instantaneous graph G_4

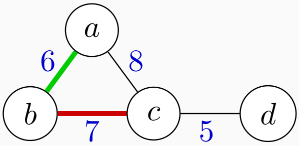


({b, c}, [4, 7]), ({a, b, c}, [4, 6]), ~~({a, b}, [4, 6])~~

- $\rightarrow$ cliques of G_4 containing a new edge
- $\rightarrow$ starting time: $t = 4$
- $\rightarrow$ end time: min of edge end times
- $\rightarrow$ filter vertex-maximal cliques



Instantaneous graph G_4



({b, c}, [4, 7]), ({a, b, c}, [4, 6]), ~~({a, b}, [4, 6])~~

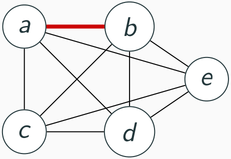
- → cliques of G_4 containing a new edge
- → starting time: $t = 4$
- → end time: min of edge end times
- → filter vertex-maximal cliques

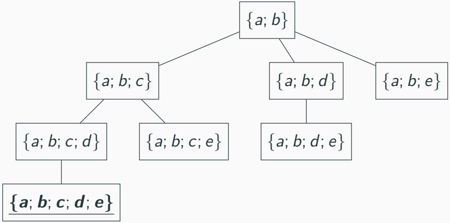
Back to a graph problem: clique enumeration in G_t.

> Pruning the enumeration in G_t

Clique enumeration

in some G_t

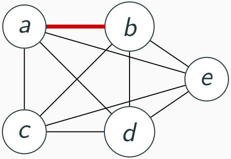


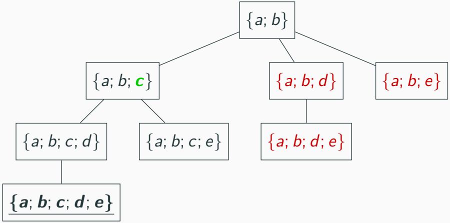


2 - New algorithm
> Pruning the enumeration in G_t

Clique enumeration

in some G_t

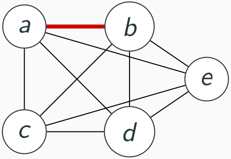


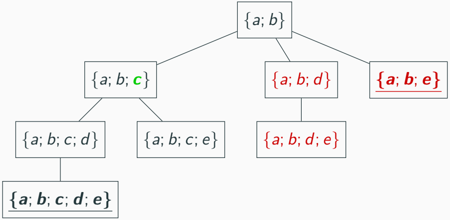


c chosen as pivot at first step

Clique enumeration

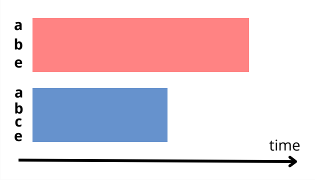
in some G_t





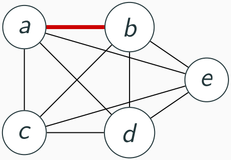
c chosen as pivot at first step

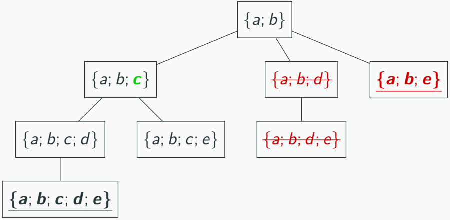
Example 1: do not prune $\{a; b; e\}$



Clique enumeration

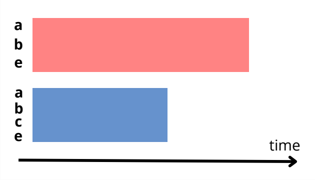
in some G_t



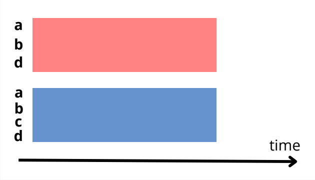


c chosen as pivot at first step

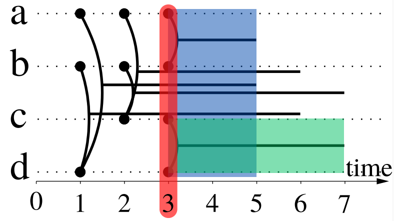
Example 1: do not prune $\{a; b; e\}$



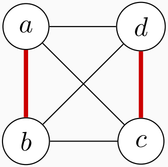
Example 2: prune $\{a; b; d\}$

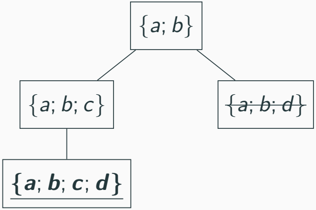


2 - New algorithm
> Pruning branches with already processed edges



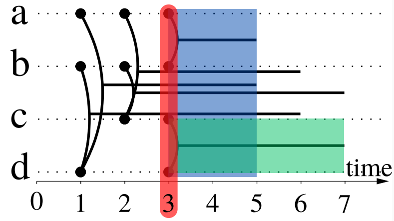
Instantaneous graph G_3



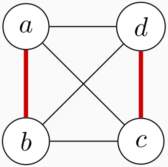


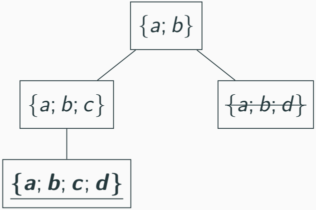
2 - New algorithm

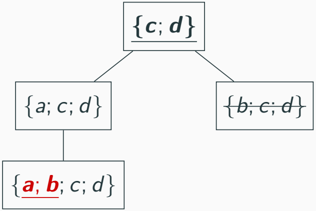
> Pruning branches with already processed edges



Instantaneous graph G_3

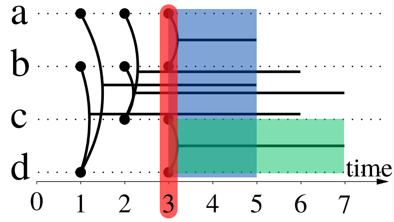




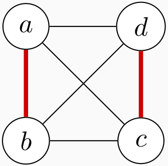


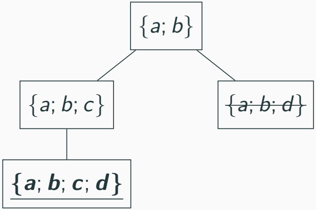
2 - New algorithm

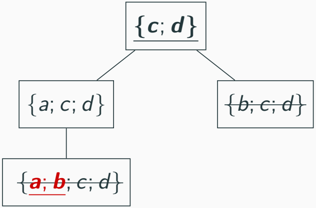
> Pruning branches with already processed edges



Instantaneous graph G_3







Summary of the new algorithm

$\rightarrow$ cliques not stored in memory ✓

$\rightarrow$ interactions reduced at each time step ✓

Summary of the new algorithm

→ cliques not stored in memory ✓

→ interactions reduced at each time step ✓

State of the art : four main works

- Viard *et al.* 2016
- Viard *et al.* 2018

Store all cliques
⇒ too costly in memory

- Himmel *et al.* 2017
- Bentert *et al.* 2019

Need all past and future interactions when processing a vertex.

3 - Complexity analysis

3 - Complexity analysis
> From input characteristics

Input characteristics

$\boldsymbol{d}$: maximal instantaneous degree

$\boldsymbol{m}$: number of links

3 - Complexity analysis
> From input characteristics

Input characteristics

***d*:** maximal instantaneous degree

***m*:** number of links

Algorithm: for each link $(u \xrightarrow{[t_0,t_1]} v)$:

$\rightarrow$ List and process cliques in G_{t_0} that contain $\{u, v\}$

- Number of those cliques: $\mathcal{O}\left(2^d\right)$
- Cost of computing and processing: $\mathcal{O}\left(d^2\right)$

Input characteristics

$\boldsymbol{d}$: maximal instantaneous degree

$\boldsymbol{m}$: number of links

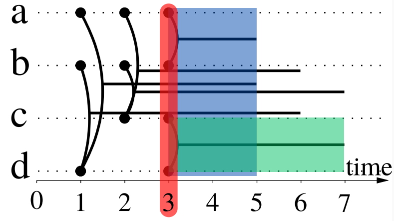
Algorithm: for each link $(u \xrightarrow{[t_0, t_1]} v)$:

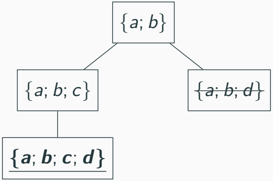
$\rightarrow$ List and process cliques in G_{t_0} that contain $\{u, v\}$

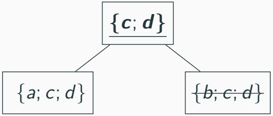
- Number of those cliques: $\mathcal{O}\left(2^d\right)$
- Cost of computing and processing: $\mathcal{O}\left(d^2\right)$

$\Rightarrow$ **Complexity:** $\boxed{\mathcal{O}\left(m \cdot d^2 \cdot 2^d\right)}$

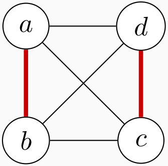
3 - Complexity analysis
> From output characteristics



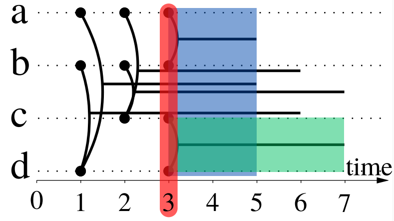


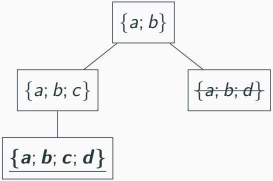


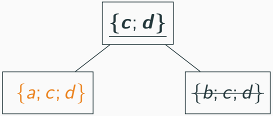
Instantaneous graph G_3



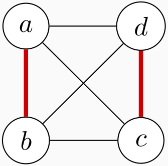
3 - Complexity analysis
> From output characteristics





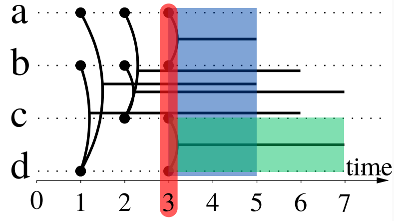


Instantaneous graph G_3

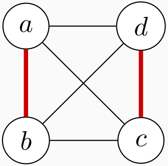


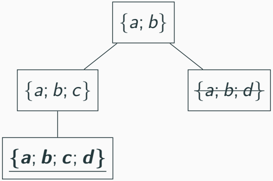
3 - Complexity analysis

> From output characteristics



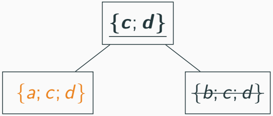
Instantaneous graph G_3





Ratio "good" leaves

$$r = \frac{\text{nb maximal clique leaves}}{\text{nb leaves}}$$



Here: $r = \frac{1}{4}$; optimal: $r = 1$.

Output characteristics

α**:** number of maximal cliques

$\boldsymbol{q}$**:** maximal number of vertices in a clique

3 - Complexity analysis
$>$ From output characteristics

Output characteristics

α**:** number of maximal cliques

$\boldsymbol{q}$**:** maximal number of vertices in a clique

Complexity: $\mathcal{O}\left((\text{nb nodes in the trees}) \cdot (\text{cost of a node})\right)$

Output characteristics

α**:** number of maximal cliques

$\boldsymbol{q}$**:** maximal number of vertices in a clique

Complexity: $\mathcal{O}\left((\text{nb nodes in the trees}) \cdot (\text{cost of a node})\right)$

$\rightarrow$ Nb nodes: $\mathcal{O}\left((\text{max depth}) \cdot (\text{nb leaves})\right)$

- max depth $= q$
- nb leaves $= \frac{1}{r} \cdot (\text{nb maximal clique leaves}) = \mathcal{O}\left(\frac{1}{r} \cdot \alpha\right)$

Output characteristics

α: number of maximal cliques

q: maximal number of vertices in a clique

Complexity: $\mathcal{O}\left((\text{nb nodes in the trees}) \cdot (\text{cost of a node})\right)$

→ Nb nodes: $\mathcal{O}\left((\text{max depth}) \cdot (\text{nb leaves})\right)$

- max depth $= q$
- nb leaves $= \frac{1}{r} \cdot (\text{nb maximal clique leaves}) = \mathcal{O}\left(\frac{1}{r} \cdot \alpha\right)$

→ Cost of a node: $\mathcal{O}\left(d^2\right)$

Output characteristics

α: number of maximal cliques

q: maximal number of vertices in a clique

Complexity: $\mathcal{O}\left((\text{nb nodes in the trees}) \cdot (\text{cost of a node})\right)$

$\rightarrow$ Nb nodes: $\mathcal{O}\left((\text{max depth}) \cdot (\text{nb leaves})\right)$

- max depth $= q$
- nb leaves $= \frac{1}{r} \cdot (\text{nb maximal clique leaves}) = \mathcal{O}\left(\frac{1}{r} \cdot \alpha\right)$

$\rightarrow$ Cost of a node: $\mathcal{O}\left(d^2\right)$

$\Rightarrow$ Complexity of the algorithm: $\boxed{\mathcal{O}\left(\frac{1}{r} \cdot \alpha \cdot q \cdot d^2\right)}$

From input characteristics

$$\boxed{\mathcal{O}\left(m \cdot d^2 \cdot 2^d\right)}$$

From output characteristics

$$\boxed{\mathcal{O}\left(\frac{1}{r} \cdot d^2 \cdot q \cdot \alpha\right)}$$

From input characteristics

$$\boxed{\mathcal{O}\left(m \cdot d^2 \cdot 2^d\right)}$$

- Viard *et al.* 2018: $\mathcal{O}\left(n^3 \cdot m^2 \cdot 2^n\right)$
- Himmel *et al.* 2018: $\mathcal{O}\left(m \cdot n \cdot |T| \cdot 3^{c/3} \cdot 2^c\right)$
- Bentert *et al.* 2019: $\mathcal{O}\left(n^4 \cdot |T|^2 \cdot 2^c\right)$

($c \leq d$ degeneracy, $|T|$ number of time steps)

From output characteristics

$$\boxed{\mathcal{O}\left(\frac{1}{r} \cdot d^2 \cdot q \cdot \alpha\right)}$$

From input characteristics

$$\boxed{\mathcal{O}\left(m \cdot d^2 \cdot 2^d\right)}$$

- Viard *et al.* 2018: $\mathcal{O}\left(n^3 \cdot m^2 \cdot 2^n\right)$
- Himmel *et al.* 2018: $\mathcal{O}\left(m \cdot n \cdot |T| \cdot 3^{c/3} \cdot 2^c\right)$
- Bentert *et al.* 2019: $\mathcal{O}\left(n^4 \cdot |T|^2 \cdot 2^c\right)$

  ($c \leq d$ degeneracy, $|T|$ number of time steps)

From output characteristics

$$\boxed{\mathcal{O}\left(\frac{1}{r} \cdot d^2 \cdot q \cdot \alpha\right)}$$

- output size $= \mathcal{O}\left(q \cdot \alpha\right) \Rightarrow$ factor $\frac{1}{r} \cdot d^2$ from output size

From input characteristics

$$\boxed{\mathcal{O}\left(m \cdot d^2 \cdot 2^d\right)}$$

- Viard *et al.* 2018: $\mathcal{O}\left(n^3 \cdot m^2 \cdot 2^n\right)$
- Himmel *et al.* 2018: $\mathcal{O}\left(m \cdot n \cdot |T| \cdot 3^{c/3} \cdot 2^c\right)$
- Bentert *et al.* 2019: $\mathcal{O}\left(n^4 \cdot |T|^2 \cdot 2^c\right)$

  ($c \leq d$ degeneracy, $|T|$ number of time steps)

From output characteristics

$$\boxed{\mathcal{O}\left(\frac{1}{r} \cdot d^2 \cdot q \cdot \alpha\right)}$$

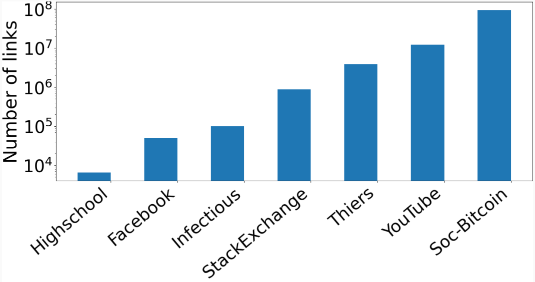
- output size $= \mathcal{O}\left(q \cdot \alpha\right) \Rightarrow$ factor $\frac{1}{r} \cdot d^2$ from output size
- $1 \leq \frac{1}{r} \leq 2^q$ but $\frac{1}{r} \approx 1.1$ in practice (experiments)

4 - Experimental study: performance gains

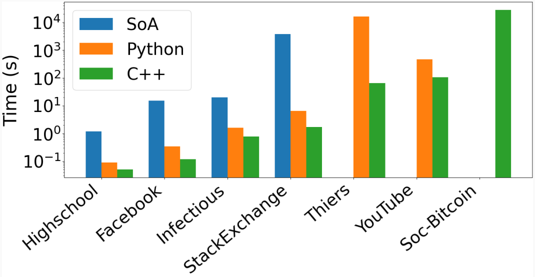
4 - Experimental study: performance gains

Datasets: state of the art + massive link streams

- communication networks
- human interactions



4 - Experimental study: performance gains



Experimental protocol

- Code in `Python` and `C++`
- Maximum 24h and 390Gb RAM.

5 - Conclusion and perspectives

5 - Conclusion and perspectives
> Contributions

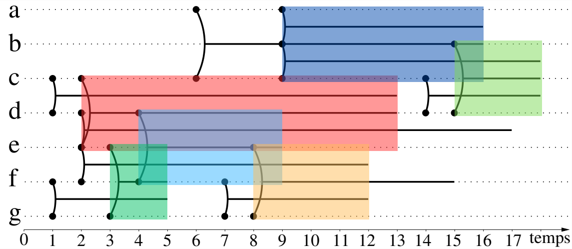
New algorithm
→ cliques not stored
→ interactions at each time step

Complexity
→ input characteristics
→ output characteristics

Experiments
→ real-world datasets
→ gain orders of magnitude:
– time computation
– size datasets

Communities in link streams by clique percolation

[Baudin et al. 2023]

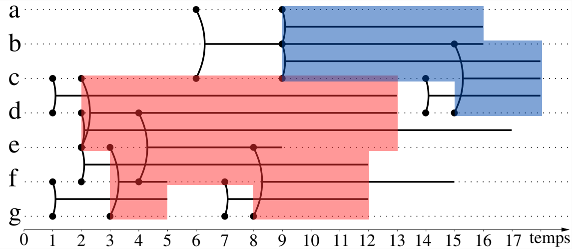


→ temporal data analysis

→ anomaly detection

Communities in link streams by clique percolation

[Baudin et al. 2023]



→ temporal data analysis

→ anomaly detection

Improve enumeration by ordering nodes

Ordering the nodes of each instantaneous graph G_t.

[Eppstein *et al.* 2010]: core ordering of vertices.

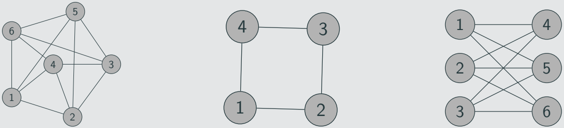
Improve enumeration by ordering nodes

Ordering the nodes of each instantaneous graph G_t.

[Eppstein *et al.* 2010]: core ordering of vertices.

Other temporal enumeration using a same framework

Enumerate other motifs in each instantaneous graph G_t.



Thank you for your attention!

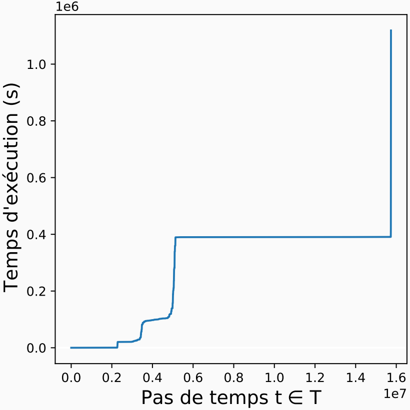
Code in open-access:

`https://gitlab.lip6.fr/baudin/maxcliques-linkstream`

Contact: `alexis.baudin@lip6.fr`

Appendix

Study the limits of computation



Study the limits of computation