Énumération efficace des cliques maximales dans les flots de liens réels massifs

Alexis Baudin, Clémence Magnien, Lionel Tabourier

Énumération efficace des cliques maximales dans les flots de liens réels massifs

Alexis Baudin*, Clémence Magnien*, Lionel Tabourier*

* Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
prenom.nom@lip6.fr

Résumé. Les flots de liens offrent un formalisme de description d'interactions au cours du temps. Un lien correspond à deux sommets qui interagissent sur un intervalle de temps. Une clique est un ensemble de sommets associé à un intervalle de temps durant lequel ils sont tous connectés. Elle est maximale si ni son ensemble de sommets ni son intervalle de temps ne peuvent être augmentés. Les algorithmes existants pour énumérer ces structures ne permettent pas de traiter des jeux de données réels de plus de quelques centaines de milliers d'interactions. Or, l'accès à des données toujours plus massives demande d'adapter les outils à de plus grandes échelles. Nous proposons alors un algorithme qui énumère les cliques maximales sur des réseaux temporels réels et massifs atteignant jusqu'à plus de 100 millions de liens. Nous montrons expérimentalement qu'il améliore l'état de l'art de plusieurs ordres de grandeur.

1 Introduction

L'analyse des réseaux d'interactions issus du monde réel a récemment fait des progrès significatifs, en passant de représentations statiques à des représentations dynamiques. La disponibilité de données temporelles, ainsi que le développement d'outils pour les décrire et les analyser, ont révélé l'importance de la temporalité des événements pour comprendre la structure et le fonctionnement de systèmes complexes en interaction, tels que les réseaux de machines, les réseaux électriques, les réseaux sociaux en ligne, ou même les réseaux biologiques.

Dans cet article, pour représenter les interactions dynamiques, nous utilisons le formalisme des flots de liens, introduit par Latapy et al. (2018). Il associe à chaque interaction un intervalle de temps, pendant lequel l'interaction existe dans le réseau. C'est un formalisme rigoureux qui permet de rendre compte à la fois des aspects temporels et structurels des données d'interaction, ce qui enrichit leur analyse, en considérant ces deux aspects intrinsèquement liés.

Le problème de l'énumération des cliques maximales dans un graphe est NP-difficile. Néanmoins, c'est un problème important pour l'analyse des graphes issus de données d'interactions réelles, car il permet de décrire leur structure. Par exemple, l'énumération de cliques est utilisée pour détecter des sous-graphes denses pertinents (Gibson et al., 2005; Fratkin et al., 2006), pour définir des communautés (Palla et al., 2005; Baudin et al., 2022), ou encore pour la compression de graphes (Buehrer et Chellapilla, 2008).

En y ajoutant une dimension temporelle, l'énumération des cliques maximales dans les flots de liens (maximales à la fois en nombre de sommets et en temps) a suscité de l'intérêt

ces dernières années, car elle apporte un puissant outil d'analyse au domaine. La détection de cliques maximales dans un flot de lien est une étape importante pour l'étude de sa structure, comme pour les graphes, il existe en effet un besoin pour des applications telles que la détection d'anomalie, elle-même utilisée pour la détection de fraudes bancaires ou d'attaques informatiques (Viard et al., 2018).

Cependant, les méthodes existantes pour l'énumération (Viard et al., 2016, 2018; Himmel et al., 2016, 2017; Bentert et al., 2019) restent limitées à des réseaux dynamiques relativement petits, et ne permettent pas le dénombrement sur des flots de liens de plus de quelques centaines de milliers de liens. Or, les avancées technologiques permettent l'accès à des jeux de données toujours plus massifs, créant un besoin d'algorithmes plus efficaces pour lister les cliques maximales.

Les contributions de cet article sont les suivantes : nous proposons un nouvel algorithme pour énumérer les cliques maximales dans les flots de liens ; nous montrons expérimentalement qu'il surpasse significativement l'état de l'art et permet d'augmenter la taille des jeux de données de deux ordres de grandeur ; nous en fournissons deux implémentations [1] : une en `Python`, pour se comparer à l'état de l'art, et une en `C++`, qui est la plus efficace actuellement disponible.

Le reste de ce papier est organisé comme suit. La Section 2 donne les définitions et notations de base que nous utilisons. La Section 3 présente les travaux de la littérature associés à l'énumération des cliques maximales dans les flots de liens. Notre nouvel algorithme est présenté dans la Section 4. Enfin, dans la Section 5, nous faisons une évaluation expérimentale des performances de notre algorithme.

2 Définitions et notations

Cliques dans un graphe. Nous rappelons quelques définitions sur les graphes. Un *graphe* $G = (V, E_G)$ est une paire constituée d'un ensemble V de *sommets*, et d'un ensemble E_G d'*arêtes*, où les arêtes sont de la forme $\{x, y\}$, avec $x, y \in V$ et $x \neq y$. Pour un sommet donné $u \in V$, le *voisinage* de u, noté $\Gamma_G(u)$, est l'ensemble des sommets adjacents à u dans G. Une *clique* C de G est un ensemble de sommets tous connectés entre eux, c'est-à-dire tels que $\forall u, v \in C$ *avec* $u \neq v, \{u, v\} \in E_G$. Elle est *maximale* si elle n'est incluse dans aucune autre.

Cliques dans un flot de liens. Nous donnons à présent des définitions autour des *flots de liens*, suivant le formalisme proposé dans Latapy et al. (2018). Dans un flot de liens, un lien entre deux sommets u et v est associé à un intervalle de temps d'existence $[b, e]$:

Définition 1 (Flot de liens)**.** *Un* flot de liens *est un triplet* $L = (T, V, E)$ *où* T *est un intervalle de temps,* V *un ensemble de sommets et* $E \subseteq T \times T \times V \times V$ *un ensemble de liens tel que pour tous les liens* (b, e, u, v) *dans* E *on a* $e \geq b$*. Nous appelons* $e - b$ *la durée du lien.*

Les flots de liens que nous utilisons sont non orientés, *i.e.* il n'y a pas de distinction entre $(b, e, u, v) \in E$ et $(b, e, v, u) \in E$. Ils sont également simples, *i.e.* pour tous les liens différents, mais sur les mêmes sommets (b, e, u, v) et (b', e', u, v), on a $[b, e] \cap [b', e'] = \emptyset$. Pour éviter toute confusion, nous utilisons le terme *arête* pour les éléments de E_G dans un graphe $G = (V, E_G)$, et *lien* pour les éléments de E dans un flot de liens $L = (T, V, E)$.

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

Définition 2 (Clique d'un flot de liens)**.** *Une* clique d'un flot de liens *est une paire* $(C, [t_0, t_1])$*, où* $t_0, t_1 \in T$ *sont respectivement appelés les temps de début et de fin de la clique, et* $C \subseteq V$ *est l'ensemble des sommets de la clique, avec* $|C| \geq 2$*. Chaque paire de sommets de* C *est reliée par un lien qui existe sur tout l'intervalle* $[t_0, t_1]$*.*

Formellement $(C, [t_0, t_1])$ *est telle que :*

$$\forall u, v \in C \text{ avec } u \neq v,\ \exists (b, e, u, v) \in E \text{ tel que } [t_0, t_1] \subseteq [b, e].$$

Notez que par souci de simplicité, nous utilisons le terme *clique* à la fois pour désigner une clique C dans un graphe et une clique $(C, [t_0, t_1])$ dans un flot de liens, alors que ces objets sont de nature différente, le contexte permettra de lever l'ambiguïté. Enfin, comme dans un graphe, les cliques contiennent des sous-cliques : nous n'énumérons que celles qui sont maximales. La notion de maximalité se traduit en termes de *temps* et de *sommets*, et elle est formalisée par les définitions suivantes. La Figure 1, à gauche, montre un flot de liens, avec ses cliques maximales en couleur.

Définition 3 (Clique maximale en temps)**.** *Une clique* $(C, [t_0, t_1])$ *est* maximale en temps *si elle ne peut pas être étendue dans le temps : il n'existe pas de clique* $(C, [t'_0, t'_1])$ *avec* $[t_0, t_1] \subsetneq [t'_0, t'_1]$*.*

Définition 4 (Clique maximale en sommets)**.** *Une clique* $(C, [t_0, t_1])$ *est* maximale en sommets *si elle ne peut pas être étendue en sommets : il n'existe pas de clique* $(C', [t_0, t_1])$ *avec* $C \subsetneq C'$*.*

Définition 5 (Clique maximale)**.** *Une clique est* maximale *si elle est à la fois maximale en temps et en sommets.*

Graphes instantanés d'un flot de liens. Nous nous intéressons à présent à la manière d'analyser le flot de liens $L = (T, V, E)$ à un temps $t \in T$ donné. L'ensemble des liens de E qui existent à t peut être vu comme l'ensemble des arêtes d'un graphe, que nous appelons le *graphe instantané de* L *au temps* t. Ce graphe contient les sommets de V et les arêtes qui existent au temps t, ou plus formellement :

Définition 6 (Graphe instantané G_t)**.** *Le* graphe instantané de L au temps $t \in T$ *est le graphe* $G_t = (V, E_{G_t})$ *tel que :*

$$E_{G_t} = \{\{u, v\} \mid \exists (b, e, u, v) \in E,\ t \in [b, e]\}.$$

Ainsi, chaque arête de G_t est induite par un lien de L qui existe à t. Ce lien a un temps de fin et nous formalisons alors les notions de *temps de fin d'une arête* et de *temps final d'une clique* de G_t :

Définition 7 (Temps de fin $\mathcal{E}_t(u, v)$ d'une arête $\{u, v\}$ de E_{G_t})**.** *Soit* $\{u, v\}$ *une arête de* E_{G_t}*. Par définition de* G_t*, il existe un lien* $(b, e, u, v) \in E$ *tel que* $t \in [b, e]$*. Nous appelons* e *le* temps de fin de l'arête $\{u, v\}$ *dans* G_t*, et nous le notons* $\mathcal{E}_t(u, v)$*.*

Définition 8 (Temps final $\mathcal{E}_t(C)$ d'une clique C de G_t)**.** *Soit* C *une clique de* G_t*. Le* temps final de la clique C*, noté* $\mathcal{E}_t(C)$*, est le minimum des temps de fin de toutes les arêtes de la clique* C *sur* G_t*. Formellement :*

$$\mathcal{E}_t(C) = \min_{u,v \in C} \{\mathcal{E}_t(u, v)\}.$$

Par exemple, à droite de la Figure 1, le graphe instantané au temps $t = 4$ est le graphe $G_4 = (\{a, b, c, d\}, E_4)$, avec $E_4 = \{\{a, b\}, \{a, c\}, \{b, c\}, \{c, d\}\}$. Les temps de fin de ses arêtes sont : $\mathcal{E}_4(a, b) = 7$, $\mathcal{E}_4(a, c) = 6$, $\mathcal{E}_4(b, c) = 8$ et $\mathcal{E}_4(c, d) = 12$. G_4 contient la clique $\{a, b, c\}$, qui existe aussi dans G_5 et dans G_6, et dont le temps final est 6, qui correspond au minimum du temps de fin de chacune de ses trois arêtes.

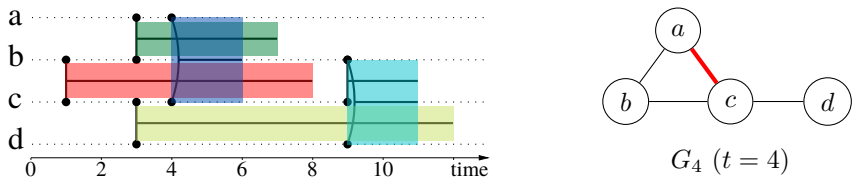


FIG. 1 – **À gauche :** *un flot de liens, avec les sommets en ordonnée et le temps d'interaction en abscisse. Par exemple, il y a un lien entre* b *et* c *sur l'intervalle de temps* $[1, 8]$*. Les cliques maximales sont représentées en couleur. Par exemple, sur l'intervalle* $[4, 6]$*, les trois sommets* a, b, c *sont reliés entre eux, et forment une clique maximale* $(\{a, b, c\}, [4, 6])$*.* **À droite :** *le graphe instantané de ce flot de liens à* $t = 4$*,* G_4*, avec en rouge l'arête qui commence à* $t = 4$*.*

3 État de l'art

Cliques maximales dans les graphes. Parmi les nombreux articles sur la recherche de cliques maximales dans les graphes, l'algorithme fondamental de Bron et Kerbosch (1973), noté BK, résout ce problème efficacement en pratique et sert de base à la conception de notre algorithme. Il s'agit d'un algorithme de retour sur trace récursif, qui est formellement décrit par l'Algorithme 1. Il maintient une clique R avec l'ensemble des voisins de tous ses sommets, qui sont partagés entre deux ensembles P et X (*i.e.* $P \cup X = \bigcap_{v \in R} \Gamma_G(v)$). P correspond aux voisins utilisés pour faire grossir la clique R, tandis que X correspond aux sommets auxquels il est interdit d'étendre R pour éviter l'énumération de cliques déjà énumérées. La clique R est maximale si et seulement si aucun sommet n'est voisin de tous ses sommets, c'est-à-dire si et seulement si $P \cup X = \emptyset$ (lignes 5 et 6).

Pour réduire la recherche, Bron et Kerbosch (1973) ont introduit l'idée d'élaguer l'arbre des appels récursifs, en sélectionnant un sommet pivot $p \in P \cup X$. L'idée est que toute clique maximale qui contient R inclut soit p soit un sommet qui n'est pas voisin de p. Alors, aucune clique maximale n'est manquée en ne faisant pas les appels récursifs sur les sommets de $\Gamma_G(p)$ (lignes 7 et 8). Les stratégies de sélection du pivot pour réaliser un élagage efficace ont été discutées dans divers travaux, par exemple Koch (2001). Actuellement, les versions les plus efficaces de cet algorithme sont inspirées du pivot proposé par Tomita et al. (2006), qui maximise $|P \cap \Gamma_G(p)|$ pour $p \in P \cup X$, et son implémentation par Eppstein et al. (2010) qui est efficace pour les grands graphes peu denses, avec une méthode adéquate d'ordonnancement des sommets. Pour calculer les cliques maximales dans les flots de liens, nous nous inspirons de cette implémentation et de ce pivot.

Cliques maximales dans les flots de liens. À notre connaissance, il existe trois principaux travaux pour énumérer les cliques maximales dans les flots de liens : Viard et al. (2016, 2018);

Algorithm 1 Algorithme de Bron-Kerbosch (avec pivot) sur un graphe G.

```
1: Entrée : G ← (V, E) graphe
2: Sortie : les cliques maximales de G
3: BK(∅,V,∅)                                   ▷ Lancement de la procédure
4: function BK(R, P, X)
5:     if P ∪ X == ∅ then
6:         output R clique maximale
7:     Choisir un pivot p ∈ P ∪ X
8:     for u ∈ P \ Γ_G(p) do
9:         BK(R ∪ {u}, P ∩ Γ_G(u), X ∩ Γ_G(u))
10:        P ← P \ {u}
11:        X ← X ∪ {u}
```

Himmel et al. (2016, 2017); Bentert et al. (2019). Notez que les flots de liens sont appelés graphes temporels dans Himmel et al. (2016, 2017) et dans Bentert et al. (2019). Tous ces travaux considèrent qu'une clique est un ensemble de sommets qui interagissent sur un intervalle de temps donné. Viard et al. (2018) montrent qu'il est simple et équivalent de passer d'une représentation à l'autre, ce qui permet de comparer les différentes méthodes.

Dans le travail original de Viard et al. (2016) les liens n'ont pas de durée, mais les cliques sont elles-mêmes paramétrées avec une valeur Δ, et sont appelées Δ-cliques. Ce paramètre est défini de telle sorte que toutes les paires de sommets dans la Δ-clique interagissent au moins une fois pendant chaque sous-intervalle de durée Δ. Leur algorithme a depuis été simplifié et étendu au formalisme plus général du flot de liens par Viard et al. (2018) ; mais il s'appuie sur de la mémoïsation afin de décider si une clique donnée a déjà été traitée, ce qui induit une forte consommation de mémoire et est prohitif dans beaucoup de cas. En parallèle, Himmel et al. (2016, 2017) ont proposé un autre algorithme pour énumérer les Δ-cliques. Ils adaptent l'algorithme BK à ce cadre dynamique, et mettent en œuvre différentes stratégies pour le choix du pivot. Cette version est plus efficace que l'algorithme de Viard et al. (2016), et l'est plus ou moins que celui de Viard et al. (2018), selon les paramètres expérimentaux et les données étudiées. Plus récemment, une généralisation de cet algorithme a été proposée par Bentert et al. (2019) pour lister tous les Δ-k-plexes maximaux dans les flots de liens. Sans définir un Δ-k-plexe, retenons qu'avec $k = 1$, c'est équivalent à une Δ-clique, ce qui permet de comparer l'algorithme aux autres décrits ci-dessus, en montrant qu'il est plus efficace.

Enfin, les interactions d'un flot de liens peuvent être représentées par une succession de graphes instantanés. Par conséquent, certaines méthodes conçues pour mettre à jour l'ensemble des cliques d'un graphe lorsqu'on lui ajoute ou supprime des arêtes peuvent être exploitées dans le contexte de l'énumération des cliques dans les flots de liens. En particulier, Das et al. (2019) appliquent la méthode d'énumération de Tomita et al. (2006) et l'utilisent pour mettre à jour les cliques d'un graphe après lui avoir ajouté un ensemble d'arêtes donné. Ceci est fait en listant les cliques qui contiennent au moins l'une des nouvelles arêtes. Bien que le problème formel soit différent du nôtre, leur méthode peut être directement adaptée à notre problème, comme nous le verrons dans la Section 4.

4 Algorithme

Notre algorithme est basé sur une adaptation de l'algorithme de Bron et Kerbosch (1973) à un cadre dynamique, en s'inspirant de ce que propose Himmel et al. (2017). La principale différence avec les travaux de l'état de l'art est que nous travaillons sur des voisinages de sommets limités aux interactions à un temps t, et non sur l'ensemble des interactions du flot de liens, ce qui permet un gain significatif d'efficacité. À noter que nous utilisons dans cette partie plusieurs théorèmes qui ne sont pas démontrés pour des contraintes de place, mais qui le sont dans une version longue de cet article en préparation. Dans ce qui suit, nous considérons un flot de liens $L = (T, V, E)$.

Structure générale de l'algorithme. L'Algorithme 2 énumère pour chaque $t \in T$ les cliques maximales qui commencent en t. Dans ce but, il énumère les cliques maximales *en temps* commençant à t (lignes 4 à 9), puis filtre celles qui sont maximales *en sommets* (lignes 10 à 12), et donc maximales. Pour énumérer les cliques maximales en temps commençant à t, nous utilisons l'équivalence du Théorème 1 : ce sont les cliques de la forme $(C, [t, \mathcal{E}_t(C)])$, où C est une clique du graphe instantané G_t contenant une arête issue d'un lien qui commence à t. Il suffit donc d'énumérer ces cliques de G_t. Cette énumération est faite par les appels à la fonction GRAPHCLIQUEENUM (ligne 8), qui est expliquée dans le prochain paragraphe, puis nous expliquons le filtrage des cliques maximales en sommets dans le paragraphe d'après. Nous résumons la structure générale de l'algorithme par la Figure 2. Pour illustrer son fonctionnement sur l'exemple de la Figure 1, nous voyons que les cliques énumérées de G_4 sont celles qui contiennent $\{a, c\}$: $\{a, c\}$ et $\{a, b, c\}$; les cliques maximales en temps commençant à $t = 4$ sont donc $(\{a, c\}, [4, 6])$ et $(\{a, b, c\}, [4, 6])$; la première n'est pas maximale en sommets, car elle est incluse dans la deuxième, elle n'est donc pas conservée pour l'énumération.

Théorème 1 (Maximalité en temps d'une clique)**.** *$(C, [t_0, t_1])$ est une clique maximale en temps si et seulement si :cd*

(i) C est une clique de G_{t_0} ;

(ii) il existe une arête $\{u, v\}$ avec u et v dans C qui provient d'un lien $(t_0, e, u, v) \in E$ dont le temps de début est t_0 ;

(iii) $t_1 = \mathcal{E}_{t_0}(C)$.

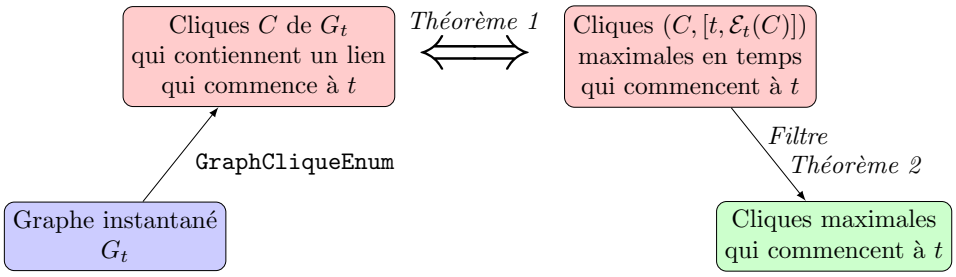


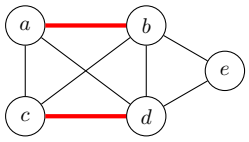
FIG. 2 – *Structure générale de l'Algorithme 2 : pour chaque temps $t \in T$, il énumère l'ensemble des cliques maximales qui commencent à t.*

Algorithm 2 Algorithme d'énumération des cliques maximales dans les flots de liens.

1: **Entrée :** $L = (T, V, E)$ flot de liens
2: **Sortie :** les cliques maximales de L
3: **for** $t \in T$ **do**
4: $\quad Cliques \leftarrow \emptyset$ ▷ Cliques qui contiennent un lien qui commence à t
5: $\quad ForbidEdges \leftarrow \emptyset$
6: $\quad$ **for** $(t, \_, u, v) \in E$ **do** ▷ Liens qui commencent à t
7: $\quad\quad P_{uv} \leftarrow \Gamma_{G_t}(u) \cap \Gamma_{G_t}(v)$
8: $\quad\quad Cliques \leftarrow Cliques \cup$ GRAPHCLIQUEENUM$(\{u, v\}, P_{uv}, \emptyset, ForbidEdges, t)$
9: $\quad\quad ForbidEdges \leftarrow ForbidEdges \cup \{\{u, v\}\}$
10: $\quad$ **for** $(C, \mathcal{N}_C) \in Cliques$ **do** ▷ $\mathcal{N}_C =$ voisins de la clique C
11: $\quad\quad$ **if** $\forall u \in \mathcal{N}_C,\ \mathcal{E}_t(C \cup \{u\}) < \mathcal{E}_t(C)$ **then** ▷ Test de maximalité en sommets
12: $\quad\quad\quad$ **output** $(C, [t, \mathcal{E}_t(C)])$
13: **function** GRAPHCLIQUEENUM(R, P, X, $ForbidEdges$, t)
14: $\quad$ **output** $(R, P \cup X)$ ▷ $P \cup X =$ voisins de la clique R
15: $\quad Q \leftarrow \{u \in P \mid \exists \{u, v\} \in ForbidEdges,\ v \in R\}$
16: $\quad$ **for** $u \in P \setminus Q$ **do**
17: $\quad\quad$ GRAPHCLIQUEENUM($R \cup \{u\}$, $P \cap \Gamma_{G_t}(u)$, $X \cap \Gamma_{G_t}(u)$, $ForbidEdges$, t)
18: $\quad\quad P \leftarrow P \setminus \{u\}$
19: $\quad\quad X \leftarrow X \cup \{u\}$

Lister les cliques des graphes G_t avec la fonction GRAPHCLIQUEENUM. Un appel à GRAPHCLIQUEENUM$(R, P, X, ForbidEdges, t)$ (ligne 13) renvoie toutes les cliques de G_t qui contiennent R, des sommets de P et aucune arête de $ForbidEdges$. Il s'agit d'une variante de l'algorithme de Bron et Kerbosch (1973). Ici, la clique en construction est R, son voisinage est $P \cup X$, et les candidats pour augmenter R sans énumérer de doublons sont les sommets de P. Notez que les cliques sont retournées qu'elles soient maximales ou non dans G_t, et avec leur voisinage (nécessaire au test de maximalité de la ligne 11). Enfin, la ligne 15 assure qu'elles ne contiennent aucune arête de $ForbidEdges$: il est interdit d'ajouter un sommet u à R si cela implique l'ajout d'une arête $\{u, v\}$ de $ForbidEdges$ à la clique résultante $R \cup \{u\}$. L'ensemble d'arêtes $ForbidEdges$ permet ainsi d'éviter d'énumérer plusieurs fois une même clique, à l'image de ce que proposent Das et al. (2019). Par exemple, dans le graphe instantané de la Figure 3, la clique $\{a, b, c, d\}$ contient deux nouvelles arêtes : $\{a, b\}$ et $\{c, d\}$. Alors, pour ne pas qu'elle soit énumérée deux fois, lorsque les cliques qui contiennent l'arête $\{a, b\}$ sont énumérées par la ligne 8, l'arête $\{a, b\}$ est ajoutée à l'ensemble $ForbidEdges$ à ligne 9. Ainsi, la clique $\{a, b, c, d\}$ ne peut pas être énumérée par l'appel qui énumère les cliques contenant $\{c, d\}$, et donc elle n'est énumérée qu'une fois.

Filtrer les cliques maximales en sommets parmi les cliques maximales en temps. La ligne 11 filtre les cliques maximales en sommets parmi l'ensemble des cliques maximales en temps qui commencent à t. Pour cela, elle utilise l'équivalence du Théorème 2 : une clique $(C, [t, \mathcal{E}_t(C)])$ qui est maximale en temps est maximale en sommets (et donc maximale) si et seulement si on ne peut lui ajouter aucun sommet de son voisinage sans réduire strictement son temps final. Son voisinage correspond aux voisins de l'ensemble de ses sommets dans G_t,



Un graphe instantané G_t

FIG. 3 – *Exemple d'une énumération de cliques utilisant $ForbidEdges$. Un graphe instantané G_t est représenté, pour $t \in T$ donné. Les arêtes rouges épaisses correspondent à des liens qui commencent à t, tandis que les autres correspondent à des liens qui ont commencé plus tôt.*

i.e. $\bigcap_{v \in C} \Gamma_t(c)$, noté $\mathcal{N}_C$ à la ligne 11. Par exemple, dans la Figure 1, à $t = 4$, $(\{a, c\}, [4, 6])$ n'est pas maximale en sommets, car on peut lui ajouter le sommet b, voisin de a et de c dans G_4, sans réduire son temps final. Notez que si $\mathcal{N}_C$ est vide, alors $(C, [t, \mathcal{E}_t(C)])$ est toujours maximale en sommets.

Théorème 2 (Maximalité en sommets d'une clique maximale en temps)**.** *Soit $(C, [t, \mathcal{E}_t(C)])$ une clique maximale en temps. Alors :*

$$(C, [t, \mathcal{E}_t(C)]) \textit{ est maximale en sommets } \Leftrightarrow \forall u \in \bigcap_{v \in C} \Gamma_t(v),\ \ \mathcal{E}_t(C \cup \{u\}) < \mathcal{E}_t(C).$$

Stratégie du pivot pour améliorer l'énumération des cliques dans les graphes G_t. Nous adaptons la stratégie classique du pivot, mentionnée dans la Section 3, à l'Algorithme 2, dans la fonction GRAPHCLIQUEENUM :

Remplacer la ligne 16 par :

$p \leftarrow$ pivot $\in P \cup X$
$Del \leftarrow \{u \in P \cap \Gamma_{G_t}(p) \mid \mathcal{E}_t(R \cup \{u\}) = \mathcal{E}_t(R \cup \{u, p\})\}$
for $u \in P \setminus Del \setminus Q$ **do** ...

Le pivot retire les sommets de Del des candidats à l'augmentation de R. On peut montrer que l'élimination de ces sommets n'élimine de l'énumération aucune clique maximale du flot de liens (non-détaillé ici). Pour sélectionner le pivot $p \in P \cup X$, nous nous inspirons des travaux de Tomita et al. (2006) et Himmel et al. (2017), et en choisissons un qui maximise le nombre de sommets à éliminer des appels récursifs : $|Del|$.

5 Expériences

Implémentations et matériel. Nos expériences sont réalisées sur une machine équipée de 2 processeurs Intel Xeon Silver 4216 de 32 cœurs chacun, et de 384 Go de RAM. Nous expérimentons ici l'Algorithme 2 avec pivot, implémenté en deux langages : en `Python` pour nous comparer à l'état de l'art qui est lui-même implémenté en `Python`, et en `C++` pour permettre le calcul sur des flots de liens massifs [2].

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

Jeux de données. En suivant les travaux de Viard et al. (2018), nous construisons nos jeux de données à partir de flots de liens instantanés dans lesquels tous les liens ont une durée égale à 0, et nous associons une durée Δ à chaque lien (t, u, v) en le remplaçant par $(t, t + \Delta, u, v)$. Cela permet, d'une part, de se comparer à l'état de l'art qui étudie les Δ-cliques dans les flots de liens instantanés, puisque Viard et al. (2018) ont montré que les deux problèmes sont équivalents ; d'autre part, d'étudier l'impact de Δ sur l'efficacité de l'algorithme et les résultats retournés. En effet, ce paramètre modifie l'échelle de temps de l'analyse. Par exemple, dans un réseau d'interactions humaines, un Δ d'une journée masque la complexité de la dynamique des interactions quotidiennes, mais rend compte des différences entre interactions durant la semaine et au cours du week-end, dont on sait qu'elles ne correspondent pas aux mêmes cercles sociaux en général.

Nous utilisons les jeux de données de la référence la plus récente (Bentert et al., 2019), ainsi que des flots de liens massifs issus de données réelles (Paranjape et al., 2017; Mislove, 2009; Rossi et Ahmed, 2015; Isella et al., 2011). Ces données sont issues d'échanges de paquets dans des réseaux de communication, d'interactions humaines en ligne ou physique, et d'interactions biologiques. Dans les jeux de données issus de Bentert et al. (2019), les valeurs de Δ sont choisies identiques à celles utilisées dans cet article, à des fins de comparaison. Dans le jeu de données massif, les valeurs de Δ sont choisies comme des fonctions de Θ, la durée totale du flot de liens : soit 0, soit $\Theta/10000$, soit $\Theta/100$. Ces jeux de données sont peu denses, bien que le degré maximal peut être élevé, allant jusqu'à 36 000, le degré moyen reste faible, et les tailles des cliques ne dépassent pas quelques dizaines (allant jusqu'à 237 pour le flot de liens le plus massif).

Résultats. Nous comparons notre implémentation `Python` à celles disponibles dans l'état de l'art (également programmées en `Python`) issues de Viard et al. (2018) [3], Himmel et al. (2016) [4], et Bentert et al. (2019) [5]. La Figure 4 illustre les résultats des expériences sur l'ensemble des jeux de données, où les temps sont donnés en secondes. Les flots de liens y sont triés en abscisse du plus petit au plus massif (on notera que les échelles sont logarithmiques, pour voir les différents ordres de grandeur). Le Tableau 1 montre les résultats de manière plus détaillée sur un échantillon de nos jeux de données, choisi pour couvrir différents ordres de grandeurs du nombre de liens et de cliques. On observe que dans tous les cas testés, notre implémentation `Python` (triangles bleus) est plus efficace que l'état de l'art (croix vertes), et que plus la taille du flot de liens augmente, plus le gain est important. Sur la plupart des cas où les implémentations de l'état de l'art terminent, il y a un gain de temps d'au moins un facteur 10, et allant jusqu'à 10^3. D'autre part, au-delà de $m = 10^6$, ces implémentations ne terminent pas dans les limites de temps et de mémoire du protocole, alors que notre implémentation `Python` produit des résultats pour une partie de ces flots de liens.

Par ailleurs, nous observons un gain significatif entre l'implémentation `C++` (disques rouges) par rapport à notre implémentation `Python` (triangles bleus). Sur les jeux de données de l'état de l'art, le temps de calcul de cette implémentation ne dépasse jamais 3 secondes. Sur les flots de liens massifs (de plus de 1 million de liens), pour lesquels l'état de l'art ne produit pas de résultat dans les limites de temps et de mémoire du protocole, l'implémentation `C++`

3. `https://bitbucket.org/tiph_viard/cliques`
4. `https://fpt.akt.tu-berlin.de/temporalcliques/`
5. `https://fpt.akt.tu-berlin.de/temporalkplex/`

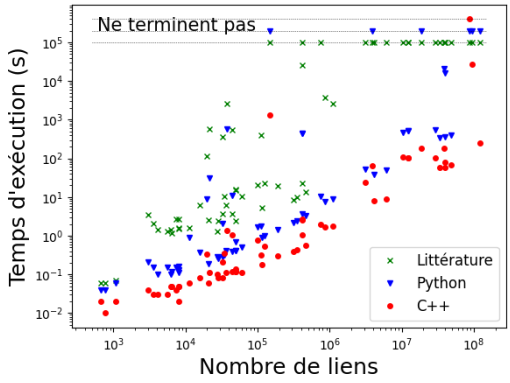


FIG. 4 – *Temps d'exécution en secondes en fonction du nombre de liens du flot de liens, sur l'ensemble des jeux de données. Le temps "Littérature" correspond au meilleur temps parmi les implémentations de l'état de l'art. Les lignes en haut de la figure correspondent aux exécutions pour lesquelles le calcul ne termine pas, stoppées au bout de 24 heures ou 380 Go de RAM.*

Flot de liens	Δ	m	α	**C++**	**Py**	**BHM+**	**VML**	**HMNS**
highschool-2011	125	6 472	7 732	0.05	0.12	1.2	1.6	5.4
facebooklike	3 125	34 116	34 342	0.35	0.31	25	10	59
facebooklike	125	50 056	50 080	0.12	0.42	26	15	96
infectious	125	100 329	138 670	0.78	1.7	634	20	945
stackexchange	23 961	870 128	894.317	1.7	7.5	-	3 747	-
youtube	1 944	12 223 774	12 253 571	104	513	-	×	-
wikipedia	19 318	39 246 821	40 898 684	79	16 223	-	×	-
soc-bitcoin	15 653	93 897 987	787 519 128	27 660	-	-	×	-
soc-bitcoin	1 565 366	86 668 193	-	-	-	-	×	-

TAB. 1 – *Résultats obtenus sur un échantillon des jeux de données. Δ correspond à la durée associée aux liens, m au nombre de liens, α au nombre de cliques. Les temps sont en secondes, et représentent les temps d'exécution de nos implémentations (***C++*** pour le* `C++` *et* **Py** *pour le* `Python`*), et celles de l'état de l'art (***BHM+*** pour Bentert et al. (2019), **VML** pour Viard et al. (2018) et **HMNS** pour Himmel et al. (2017)). Un "-" correspond à un calcul qui ne termine pas au bout de 24 heures, et un "×" qui ne termine pas à moins de 380 Go RAM.*

termine pour tous les flots de liens excepté 1, et cette implémentation permet une énumération des cliques en moins de 5 minutes pour tous les jeux de données du protocole exceptés 3. Elle permet donc de faire passer à l'échelle l'énumération des cliques maximales aux flots de liens massifs non traités par l'état de l'art.

6 Conclusion

Dans cet article, nous avons traité du problème de l'énumération des cliques maximales dans les flots de liens : nous avons proposé un nouvel algorithme qui permet de passer à l'échelle de réseaux temporels massifs, puis nous avons réalisé un protocole expérimental

sur des jeux de données variés issus d'interactions réelles, dans le but de comparer les performances de notre algorithme à l'état de l'art, et montré que nous pouvions obtenir un gain de performance de plusieurs ordres de grandeur. Les preuves de correction de l'algorithme proposé, ainsi que des considérations théoriques sur la complexité seront détaillées dans une version étendue et peuvent être mise à disposition des lecteurs intéressés.

Pour poursuivre ce travail, il serait intéressant d'adapter cet algorithme à l'énumération de cliques de taille fixée dans les flots de liens, car cette tâche est une étape utile à la description des jeux de données d'interactions. En particulier, la détection de sous-flots de liens denses est pertinente pour définir des notions de communautés dans les réseaux d'interactions, un concept notamment utilisé pour la détection d'anomalies.

Remerciements

Ce travail est financé par l'ANR (Agence Nationale de la Recherche) à travers l'ANR FiT LabCom. Les auteurs remercient la communauté SocioPatterns [6] pour la mise à disposition de leurs jeux de données.

Références

Baudin, A., M. Danisch, S. Kirgizov, C. Magnien, et M. Ghanem (2022). Clique percolation method : memory efficient almost exact communities. In *Proceedings of ADMA'22*, pp. 113–127. Springer.

Bentert, M., A.-S. Himmel, H. Molter, M. Morik, R. Niedermeier, et R. Saitenmacher (2019). Listing all maximal k-plexes in temporal graphs. *Journal of Experimental Algorithmics (JEA) 24*, 1–27.

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

Buehrer, G. et K. Chellapilla (2008). A scalable pattern mining approach to web graph compression with communities. In *Proceedings of WSDM'08*, pp. 95–106.

Das, A., M. Svendsen, et S. Tirthapura (2019). Incremental maintenance of maximal cliques in a dynamic graph. *The VLDB Journal 28*(3), 351–375.

Eppstein, D., M. Löffler, et D. Strash (2010). Listing all maximal cliques in sparse graphs in near-optimal time. In *Proceedings of ISAAC'10*, pp. 403–414. Springer.

Fratkin, E., B. T. Naughton, D. L. Brutlag, et S. Batzoglou (2006). Motifcut : regulatory motifs finding with maximum density subgraphs. *Bioinformatics 22*(14), e150–e157.

Gibson, D., R. Kumar, et A. Tomkins (2005). Discovering large dense subgraphs in massive graphs. In *Proceedings of VLDB'05*, pp. 721–732.

Himmel, A.-S., H. Molter, R. Niedermeier, et M. Sorge (2016). Enumerating maximal cliques in temporal graphs. In *Proceedings of ASONAM'16*, pp. 337–344. IEEE.

6. `http://www.sociopatterns.org`

Himmel, A.-S., H. Molter, R. Niedermeier, et M. Sorge (2017). Adapting the bron–kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining 7*(1), 1–16.

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

Koch, I. (2001). Enumerating all connected maximal common subgraphs in two graphs. *Theoretical Computer Science 250*(1-2), 1–30.

Latapy, M., T. Viard, et C. Magnien (2018). Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining 8*(1), 1–29.

Mislove, A. (2009). *Online Social Networks : Measurement, Analysis, and Applications to Distributed Information Systems*. Ph. D. thesis, Rice University, Department of Computer Science.

Palla, G., I. Derényi, I. Farkas, et T. Vicsek (2005). Uncovering the overlapping community structure of complex networks in nature and society. *nature 435*(7043), 814–818.

Paranjape, A., A. R. Benson, et J. Leskovec (2017). Motifs in temporal networks. In *Proceedings of WSDM'17*, WSDM '17, New York, NY, USA, pp. 601–610. ACM.

Rossi, R. A. et N. K. Ahmed (2015). The network data repository with interactive graph analytics and visualization. In *AAAI*.

Tomita, E., A. Tanaka, et H. Takahashi (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical computer science 363*(1), 28–42.

Viard, T., R. Fournier-S'Niehotta, C. Magnien, et M. Latapy (2018). Discovering patterns of interest in ip traffic using cliques in bipartite link streams. In *International Workshop on Complex Networks*, pp. 233–241. Springer.

Viard, T., M. Latapy, et C. Magnien (2016). Computing maximal cliques in link streams. *Theoretical Computer Science 609*, 245–252.

Viard, T., C. Magnien, et M. Latapy (2018). Enumerating maximal cliques in link streams with durations. *Information Processing Letters 133*, 44–48.

Summary

Link streams provide a consistent formalism to represent interactions in time. A link consists of two vertices that interact over a given time interval. With this formalism, a clique corresponds to a set of vertices and a time interval during which they are all linked together. It is maximal if neither its set of vertices nor its time interval can be increased. Existing algorithms for enumerating these structures are not able to handle datasets with more than a few hundred thousand links. However, access to increasingly massive data requires tools to be adapted to larger scales. We then propose an algorithm that scales up the enumeration of maximal cliques to massive real-world temporal networks of up to more than 100 million links. We show experimentally that this algorithm is more efficient than the state-of-the-art by several orders of magnitude.