Université de Mons
Faculté des sciences
Département d'Informatique

Étude comparative d'algorithme pour énumérer les cliques maximales d'un graphe

Rapport préliminaire de projet de Master

Directeur :
Hadrien Mélot
Co-directeur :
Sébastien Bonte

Auteur :
Virgil Surin

Année académique 2022-2023

Table des matières

1 Introduction

Dans ce rapport nous allons étudier plusieurs algorithmes d'énumeration de cliques maximales dans un graphe simple non orienté. Cette étude ce base sur l'article de Alessio Conte et Etsuji Tomita *"On the overall and delay complexity of the CLIQUES and Bron-Kerbosch algorithms"*.

L'objectif de ce projet est d'implémenter les algorithmes présentés et de confirmer les résultats obtenu par M.Conte et M.Tomita. Également, ces algorithmes seront implémentés en python et en rust afin ~~de pouvoir comparer leur performance d'un langage à un autre.~~

Étudier un tel problème et surtout trouver des algorithmes efficaces pour en obtenir la réponse n'est pas sans intérêt. En effet, beaucoup de systèmes peuvent être représentés par des graphes. L'on se rend alors compte que des amats se forment, ce sont là des cliques. Il s'avère que ces cliques peuvent être considérées, selon le système étudié, comme des compartiments indépendant, peut-être des unités logiques. d'où l'utilité des les trouver efficacement. Nous pouvons observer de multiples applications des cliques dans des domaines tels que la biologie, la sociologie et bien d'autres [1].

Nous commencerons par un rappel des notions et notations utilisées ainsi que des définitions utiles à la compréhension de ce rapport, suivi d'une présentation des différents algorithmes implémentés et pour terminer par la comparaison d'efficacité en temps et en langage de ceux-ci.

2 Les graphes

2.1 Notions de bases

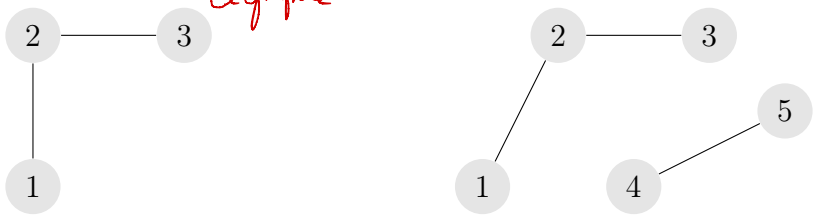
Dans ce rapport, nous allons définir un graphe simple non orienté $G(V, E)$ comme étant un ensemble de nœuds V reliés par des arêtes. Nous notons l'ensemble des arêtes comme étant l'ensemble E. Une arête est définie comme un tuple (v, v') où v et v' sont les 2 nœuds réliés par l'arête. Les nœuds peuvent être étiquetés pour plus de lisibilité.

Il y a au plus une et une seule arête entre deux ~~même~~ nœuds et aucune arête allant d'un nœud à lui-même. Les arêtes que nous considérons n'ont pas de sens particulier. Ainsi, l'arête joignant les nœuds x et y est la même que celle joignant y à x et sera dénotée par l'existence du tuple $(x, y) \in E$. Nous notons les voisins d'un nœud v par $N(v)$.

Nous définissons l'ordre ~~x~~ n d'un graphe comme étant $|V|$ et sa taille m

comme $|E|$.

Dans la figure 1, nous pouvons observer un premier graphe a dont l'ensemble des noeuds V est l'ensemble $\{1, 2, 3\}$. L'ensemble des arrêtes E quant à lui est l'ensemble $\{(1, 2), (2, 3)\}$. Le graphe a est un graphe d'ordre 3 et de taille 2. De la même façon, pour le graphe b on a $V = \{1, 2, 3, 4, 5\}$ et $E = \{(1, 2), (2, 3), (4, 5)\}$. b est d'ordre 5 et de taille 3.



(a) (b)

FIGURE 1 – Deux graphes (a et b)

Soit $W \subseteq V$ et $E(W) = E \cap (W \times W)$. Nous appellons $SG(W, E(W))$ un sous-graphe de G. Par exemple, en prenant le graphe b de la figure 1, le sous-graphe $SG(\{1, 2, 5\}, E(\{1, 2, 5\}))$ est un sous-graphe valide de b contenant uniquement les noeuds 1, 2 et 5 (W) et l'arrête $(1, 2)$ ($E(W)$).

2.2 Les cliques

Une *clique* est définie comme suit : Soit un graphe $G(V,E)$ ~~où V est l'ensemble des noeuds de G et E l'ensemble des arrêtes de G~~. Nous dirons que $C \subseteq V$, un sous-ensemble de noeuds du graphe, est une clique si et seulement si tous les noeuds de C sont voisins entre-eux. C'est-à-dire :

$$\forall v, w \in C \mid (v, w) \in E \text{ avec } v \neq w \quad (1)$$

Il est important de remarquer qu'avec cette définition d'une clique, un noeud seul n'est pas une clique.

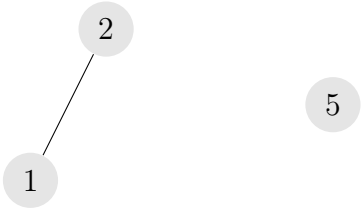


FIGURE 2 – un sous-graphe de b

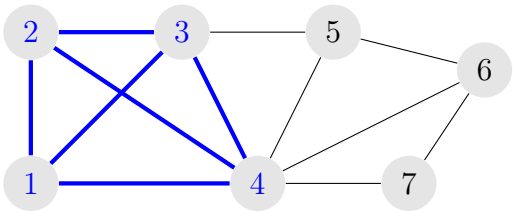


FIGURE 3 – Un graphe, en bleu une clique de ce graphe

La *taille* d'une clique est définie par sa cardinalité, c'est à dire que la taille d'une clique C est $|C|$. Comme exemple, prenons la clique en bleu dans le graphe de la figure 3 : nous avons que $C = \{1, 2, 3, 4\}$ et $|C| = 4$. C'est donc une clique de taille 4.

Nous dirons qu'une clique est *maximale* s'il est impossible de rajouter un nœuds de G dans la clique tel que la propriété ~~ci-dessus~~ reste respectée.

Dans la figure 3, la clique *bleue* composée des noeuds 1, 2, 3 et 4 est également maximale. Par contre, la clique composée des noeuds 3 et 5 ne l'est pas car il est possible d'y rajouter le noeud 4.

Enfin, nous définirons une clique *maximum* comme étant ~~la~~ clique de ~~plus grande~~ taille. Notons que toute clique maximum sera maximale et qu'il peut y avoir plusieurs cliques maximum dans un même graphe.

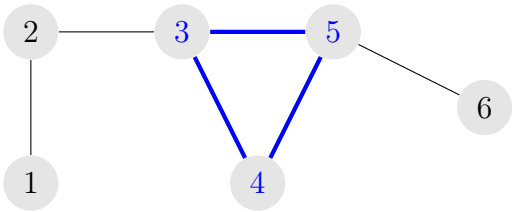


FIGURE 4 – Un graphe

Regardons la figure 4. $\{3, 4, 5\}$ (en bleu) est une clique maximale et maximum car il n'est pas possible de rajouter un noeud à cette clique tel qu'il sera voisin de 3, 4 et 5 (maximale) et il n'existe pas de clique de plus grande taille (maximum). Par contre, la clique composée des noeuds 1 et 2, bien qu'étant maximale (en effet, aucun autre noeud du graphe ne peut être ajouté à cette clique) n'est pas maximum car elle ne contient que 2 noeuds là où la clique précédemment mentionnée (celle en bleu sur la figure 4) est plus grande vu qu'elle contient 3 noeuds.

3 Le problème d'énumération des cliques dans un graphe

Ce problème consiste, dans un graphe simple, à énumérer toutes les cliques **maximales** de celui-ci. L'ordre dans lequel celles-ci sont listées n'est pas important.

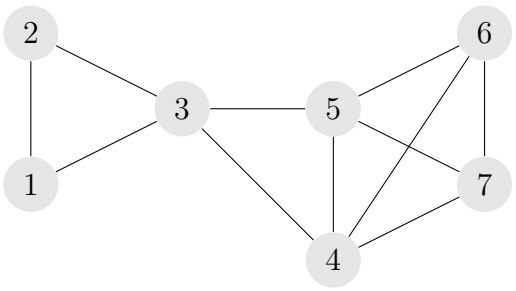


FIGURE 5

Prenons par exemple le graphe en figure 5. Nous sommes donc intéressé d'obtenir les cliques suivantes :

— $\{1, 2, 3\}$
— $\{5, 4, 3\}$
— $\{4, 5, 7, 6\}$

Ces trois cliques sont bien les seules et uniques cliques maximales de ce graphe. De plus, la clique maximum est celles contenant les noeuds 4, 5, 6 et 7. Notez comme la clique $4, 5, 6$ n'est pas reprise, en effet celle-ci n'est pas maximale car elle peut être agrandie avec le noeud 7.

3.1 Complexité du problème

Le problème d'énumération de toutes les cliques maximales dans un graphe simple est un problème appartenant à la classe NP-Complet. C'est-à-dire qu'il n'existe pas d'algorithme déterministe en temps polynomial qui permette d'énumérer toutes les cliques maximales d'un graphe.

4 Algorithmes

4.1 CLIQUE

Cet algorithme se base sur une exploration en profondeur afin d'énumérer toutes les cliques maximales de G. CLIQUE est une procédure récursive.

Conseil : 1. Idée intuitive de l'algo
2. Pseudo
3. Explications illustrées
4. Complexité

Il introduit un ensemble Q vide représentant la clique courante et essaie d'étendre Q sur base de deux sous-ensemble de noeuds $SUBG$ et $CAND$.

Initialement, $SUBG \leftarrow V$ et $CAND \leftarrow V$. L'objectif est d'énumérer toutes les cliques maximales dans le sous-graphe induit par $SUBG$ en étendant la clique courante Q avec des noeuds candidats dans $CAND$.

espaces bizarres (après aussi)

Illustrer

Soit $Q = \{p_1, p_2, ..., p_d\}$ la clique courante. Alors $SUBG = V \cap N(p_1) \cap N(p_2) \cap ... \cap N(p_d)$, c'est à dire l'ensemble des nœuds voisins de tout nœud appartenant à Q. L'algorithme selectionne alors un noeud $p \in SUBG$ et l'ajoute à Q et ceci pour tout nœud dans $SUBG$. Cela engendre un appel récursif avec $SUBG_p = SUBG \cap N(p)$ et $CANDp = CAND \cap N(p)$. p sera retiré de $CAND$ après l'exécution de l'appel récursif lors du *backtracking*.

Afin de ne pas engendrer trop d'appel récursif, nous pouvons choisir inteligemment un noeud u qui nous appélerons le *pivot*. Il existe plusieurs facon de choisir ce pivot. Conte et Tomita[2] décrivent un bon pivot u comme étant un noeud de $SUBG$ qui maximise $|CAND \cap N(u)|$. Un tel pivot va ainsi minimiser $|CAND \setminus N(u)|$, ce qui est intéressant car toute clique maximale Q' dans $SUBG \cap N(u)$ n'est pas maximale dans $G(SUBG)$ car il est possible d'ajouter u à cette clique. Ainsi, toute clique maximale contien soit u soit un noeud dans $SUBG \setminus N(u)$. Il n'est donc pas nécessaire d'explorer les voisins de u dans ce cas. Cela explique pourquoi u est un bon pivot, car maximiser $|CAND \cap N(u)|$ va minimiser $CAND \setminus N(u)$ et donc minimiser le nombre d'appel.

Ci-dessous le pseudo-code de l'algorithme décrit précédement.

Algorithm 1 Algorithme CLIQUE

Input : un graphe $G = (V, E)$
Output : toutes les cliques maximales de G

```
1: procedure CLIQUES(SUBG, CAND)
2:     if SUBG = ∅ then                          ▷ Q est une clique maximale
3:         print (Q)
4:     else
5:         u ← un noeud de SUBG qui maximise |CAND ∩ N(u)|
6:         while CAND \ N(u) ≠ ∅ do
7:             p ← un noeud dans CAND \ N(u)
8:             Q ∪ p                              ▷ on ajoute p à Q
9:             // Mise à jour des paramètres
10:            SUBG_p ← SUBG ∩ N(p)
11:            CAND_p ← CAND ∩ N(p)
12:            CLIQUES(SUBG_p, CAND_p)
13:            CAND ← CAND \ p
14:            Q \ p                              ▷ on retire p de Q
15:        end while
16:    end if
17: end procedure
18: CLIQUES(V, V)
```

La complexité dans le pire des cas de CLIQUES est en $O(3^{n/3})$[2].

Une implémentation de l'algorithme CLIQUE est disponible en python ainsi que des variantes avec d'autres pivots également présentés dans l'article.

5 Plannification

Une implémentation du code en python est déjà fournie et à l'avenir, une implémentation en Rust sera faite afin de pouvoir comparer l'exécution d'un même algorithme sur deux langages différents. En parlant de comparaison, des tests de performances sur les différents algorithmes présenter dans l'article sera réalisé afin de confirmer les résultats des auteurs de l'article en question.

D'un point de vue rédaction, il faut développer la description des différents algorithmes (et surtout tous les explicités) ainsi que de présenter les sets de données qui serviront aux tests et la méthodologie. Ceci incluant une description du matériel utilisé mais aussi la façon dont les performances vont être mesurées.

Références

[1] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[2] A. Conte and E. Tomita, "On the overall and delay complexity of the cliques and bron-kerbosch algorithms," *Theoretical Computer Science*, vol. 899, pp. 1–24, 2022.