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 algorithme 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.

Nous commencerons par un rappel des notions et notations utilisées, suivi d'une présentation des différents algorithmes implémenter pour terminer par la comparaison d'efficacité en temps de ceux-ci.

1.1 Quelques 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 arêtes 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 m d'un graphe comme étant $|V|$ et sa taille m comme $|E|$.

Soit $W \subseteq V$ et $E(W) = E \cap (W \times W)$. Nous appellons *SG(W, E(W))* un sous-graphe de G.

1.2 Définition d'une *clique*

Une *clique* C est un sous-ensemble de nœuds de V tel que tous les nœuds de C sont voisins entre-eux. C'est-à-dire :

$$\forall v, w \in C | (v, w) \in V \ avec \ v \neq w$$

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

Enfin, nous définirons une clique *maximum* comme étant la clique de plus grande cardinalité, c'est à dire contenant le plus de noeuds. Notons que toute clique maximum sera maximale et qu'il peut y avoir plusieurs cliques maximum dans un même graphe.

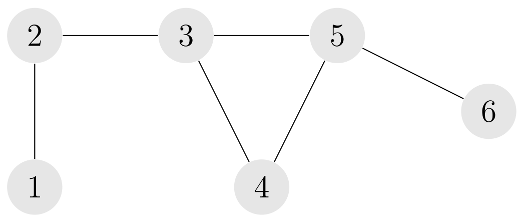

FIGURE 1 – Un graphe

Dans le graphe 1, l'ensemble $\{3, 4\}$ est une clique non maximale car il est possible d'y ajouter le noeud 5, celui-ci étant voisin de 3 et de 4.

$\{3, 4, 5\}$ 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).

2 Algorithmes

2.1 CLIQUES

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

Initialiement, $SUBG \leftarrow V$ 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*.

Soit $Q = \{v_1, v_2, ..., v_d\}$ la clique courante. Alors $SUBG = V \cap N(v_1) \cap N(v_2) \cap ... \cap N(v_d)$, c'est à dire l'ensemble des nœuds voisins de tout nœud appartenant à Q. L'algorithme selectionne alors un noeud $u \in SUBG$ et l'ajoute à Q et ceci pour tout nœud dans *SUBG*. Cela egendre un appel récursif avec $SUBG = SUBG \cap N(u)$. u sera retiré de *CAND* après l'exécution de l'appel récursif lors du *backtracking*.

Le noeud u choisi est appelé le *pivot*. Il existe plusieurs facon de choisir ce pivot. Afin de ne pas engendrer trop d'appel récursif, Conte et Tomita[1] décrivent un bon pivot u comme étant un noeud de $SUBG$ qui minimise $|CAND \cap N(u)|$

Nous ferons évoluer $SUBG$ en fonction de l'ajout ou non de u comme suit :

$$SUBG_u = SUBG \cap G(u)$$

Une clique maximale est trouvée lorsque $SUBG$ est vide. C'est à dire qu'il n'y a plus de nœud dans G qui soient voisins de tout nœud dans Q.

3 Conclusion

Références

[1] Alessio Conte et Etsuji Tomita : *"On the overall and delay complexity of the CLIQUES and Bron-Kerbosch algorithms"* - 12/11/2021