

Les cliques maximales dans les graphes dispersés

Lecture et rédaction scientifiques

Travail réalisé par Bal Sébastien
dans le cadre du Master en Informatique

Service : Master Sciences Informatiques

Directeur : Mr Mélot

Année académique 2023-2024

Table des matières

Introduction	2
Les bases et notions de vocabulaire	4
Graphes	4
Une Clique	4
Dégénérescence	5
Algorithme de Bron-Kerbosch	10
Exemple Bron-Kerbosch	10
Fonctionnement Bron-Kerbosch	13
Bron-Kerbosch Tomita et al	16
Bron-Kerbosch et la Dégénérescence	18
Conclusion	21

Introduction

Dans le cadre de cet article, nous nous penchons sur les algorithmes de recherche de cliques maximales dans les graphes, une thématique essentielle en théorie des graphes et en informatique théorique. Ces algorithmes jouent un rôle crucial dans divers domaines allant de l'analyse des réseaux sociaux à la bioinformatique, en passant par la chimie et la physique des matériaux. Notre étude se concentre particulièrement sur les contributions et les améliorations apportées aux algorithmes de Bron-Kerbosch, une méthode classique et largement utilisée pour trouver toutes les cliques maximales d'un graphe.

Le premier chapitre de cet article est consacré à l'établissement des bases nécessaires pour comprendre l'ensemble de notre travail. Il contient une présentation détaillée des notions fondamentales et du vocabulaire spécifique à l'analyse des graphes. Ce chapitre sert de guide pour les termes techniques et les concepts qui seront utilisés tout au long de cet article, garantissant ainsi une compréhension claire et approfondie des discussions ultérieures.

Dans le deuxième chapitre, nous nous concentrons sur l'algorithme original de Bron-Kerbosch. Nous y fournissons une explication détaillée de son fonctionnement, y compris les mécanismes et les étapes clés de cet algorithme. Ce chapitre vise à offrir une compréhension complète de cet algorithme classique, qui constitue la base de nos explorations et modifications subséquentes.

Le troisième chapitre introduit une amélioration notable de l'algorithme de Bron-Kerbosch, à savoir l'ajout d'un point de pivot selon l'approche de Tomita et al. Cette technique de pivotage permet d'optimiser l'algorithme en réduisant le nombre de récursions nécessaires, ce qui améliore significativement son efficacité. Nous détaillons ici le fonctionnement de cette modification et démontrons son impact sur les performances de l'algorithme.

Enfin, le dernier chapitre traite de l'algorithme BronKerboschDegeneracy, une version avancée de l'algorithme de Bron-Kerbosch qui incorpore le concept de dégénérescence des graphes. Cette approche utilise la structure de dégénérescence pour réduire davantage la complexité de la recherche de cliques maximales. Nous explorons comment cet algorithme utilise la dégénérescence pour optimiser les calculs et présenter des résultats plus rapidement et efficacement.

Au fil de la lecture de cet article, nous souhaitons découvrir les améliorations et les optimisations possibles des algorithmes de recherche de cliques maximales. Nous présenterons les concepts de base, les algorithmes initiaux et leurs évolutions, en mettant en lumière les contributions et les innovations qui ont permis de rendre ces méthodes plus performantes. En fin de parcours, le lecteur aura acquis une compréhension approfondie des algorithmes de Bron-Kerbosch et de leurs améliorations, ainsi que des techniques modernes utilisées pour optimiser la recherche de cliques maximales dans les graphes.

Les bases et notions de vocabulaire

Ce chapitre est dédié à l'explication des concepts fondamentaux et à la clarification de certains termes essentiels pour une bonne compréhension de l'article. Ces notions incluent la définition d'un graphe, l'explication des cliques, ainsi que les concepts de clique maximale et de clique maximum. La notion de dégénérescence est particulièrement importante, car elle est utilisée ultérieurement dans l'un des algorithmes présentés par les auteurs [3].

Graphes

Un *graphe* est composé d'un ensemble de sommets, ceux-ci sont reliés entre eux par des arêtes. Au niveau de la représentation, un sommet est représenté par un point et une arête par une ligne.

On donne généralement comme définition pour un graphe cette notation : $G = (V, E)$. G est un couple (V, E) dans lequel, V est un ensemble fini de sommets et E est un ensemble d'arêtes, où chaque arête est un sous-ensemble de sommets de V noté : $\{v_i, v_j\} \in V^2$.

Une arête est une entité caractérisée par une paire de sommets $\{v_i, v_j\}$. Ces deux sommets sont considérés comme adjacents. L'ensemble des sommets adjacents à un sommet $v_i \in V$ est représenté sous la notation $Adj(v_i) = \{v_i \in V, \{v_i, v_j\} \in E\}$

Une Clique

Une *clique*, au point de vue de la théorie des graphes, représente un sous-ensemble de sommets dans lequel chaque paire de sommets est reliée [5]. La *taille d'une clique* est la cardinalité d'un ensemble de k nœuds dans lequel chaque paire de nœuds est reliée par une arête. Ceux-ci sont tous connectés, on parle de *sous-graphe complet*.

Un sous-graphe est un graphe se trouvant dans un autre graphe, la figure 2 représente un sous-graphe en vert dans un graphe en bleu. Dans nos exemples,

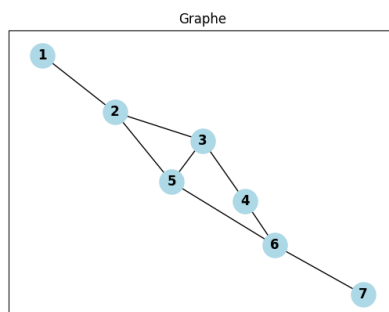


FIGURE 1 – Représentation d'un graphe

les sous-graphes mentionnés sont des sous-graphes induit, Wikipedia cite, un sous-graphe est obtenu en restreignant le graphe à un sous-ensemble de sommets et en conservant toutes les arêtes entre sommets de ce sous-ensemble [10]. Dans la suite de l'article, la notion de sous-graphe fait référence à un sous-graphe induit sauf mention contraire.

Ce qui amène à définir deux notions qui sont importantes au niveau des cliques, la clique maximale et la clique maximum. Une clique C est dite maximale s'il n'existe pas de clique plus grande contenant C [1], ce qui signifie qu'une clique maximale ne peut pas être étendue en ajoutant un autre sommet du graphe tout en maintenant sa propriété de clique. Une clique (maximale) est dite maximum si elle contient plus de sommets parmi toutes les cliques [1].

La figure 2 représente un graphe qui possède 7 sommets, les sommets en vert, numéroté 2,3,5, forment une clique de taille $k = 3$. Plusieurs méthodes existent pour détecter des cliques dans un graphe donné, celles-ci sont généralement complexes à utiliser. L'application la plus importante dans la détection de cliques est de trouver le nombre maximum de cliques dans un graphe.

Dans le cadre de ce travail, on vise à identifier et énumérer toutes les cliques maximales d'un graphe donné. Un exemple illustratif est présenté dans la figure 2, où l'on a identifié 6 cliques : les ensembles de sommets 1,2, 3,4, 4,6,5,6,6,7 et 2,3,5. Parmi ces cliques, celle qui contient le plus grand nombre de sommets est la dernière, 2,3,5, qui est une clique maximum, car elle contient trois sommets.

Dégénérescence

La *dégénérescence* dans la théorie des graphes, est un paramètre, celle-ci est la plus petite valeur de k pour qu'un graphe soit k -dégénéré. Un graphe est k -dégénéré si au moins un sommet de *degré* est supérieur à k [2]. Comme le définit Wikipedia [9], un *degré* d'un sommet dans la théorie des graphes, c'est le nombre d'arêtes reliant ce sommet.

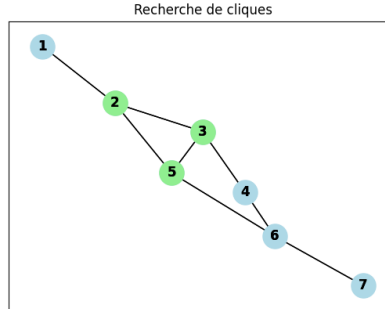


FIGURE 2 – Représentation d'une Clique

Le principe de décomposition en k -core consiste à trouver le sous-graphe le plus grand d'un réseau, dans lequel chaque nœud a au moins k voisins dans le sous-graphe. Le processus de fonctionnement pour la décomposition est que l'on supprime récursivement les nœuds ayant des degrés inférieurs à la valeur de k [4]. Dans l'article de Seidman [6], il explique qu'un k -core doit avoir au moins $k + 1$ nœuds. Pour bien comprendre, prenons l'exemple d'un groupe social, dans lequel k représente le nombre minimum de relations qu'un nœud doit avoir pour être inclus dans le k -core. Si nous avons 2-core, ce qui signifie que chaque personne dans ce 2-core a au moins 2 relations avec d'autres personnes dans le 2-core. Si nous n'avons que deux personnes dans ce 2-core. Cela signifie que chaque personne de ce 2-core a au plus 1 relation avec une autre personne dans le 2-core. Mais si chaque personne a 2 relations, cela signifie qu'il doit y avoir au moins quatre relations au total dans le 2-core. Ce qui implique qu'il doit y avoir au moins 3 personnes dans le 2-core. C'est pour cette raison qu'un k -core doit avoir au moins $k + 1$ nœuds.

Pour illustrer cela plus clairement, examinons les figures suivantes (voir les exemples de la figure 3 à 6). En ajustant la valeur de k , nous obtenons différents sous-graphes. Analysons ceux-ci de façon détaillée, le premier graphe, celui de la figure 3, est celui avec $k = 0$, cela signifie que tous les sommets peuvent être sélectionnés puisqu'il n'y a pas de contrainte au niveau du degré de dégénérescence. Les sommets qui sont sélectionnés sont en bleu.

Le graphe de la figure 4 illustre un graphe de dégénérescence avec $k = 1$. La non-conformité à la condition $k = 1$ surviendrait si un sommet n'était pas relié à un autre. Le sommet J ne respecte pas la condition de dégénérescence, il est isolé, n'ayant aucune arête incidente avec d'autres sommets. Il apparaît en rouge sur notre figure. En revanche, tous les autres sommets en bleu sont inclus dans l'ensemble de ce sous-graphe, étant donné que chaque sommet est adjacent à au moins un autre sommet.

Dans le graphe de la figure 5, avec $k = 2$, nous notons que les sommets A et J

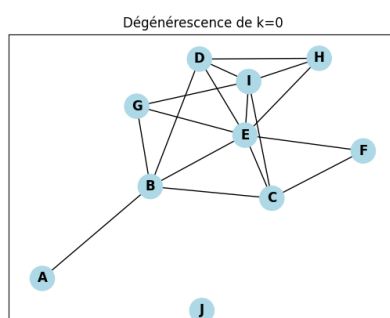


FIGURE 3 – Dégénérescence 0

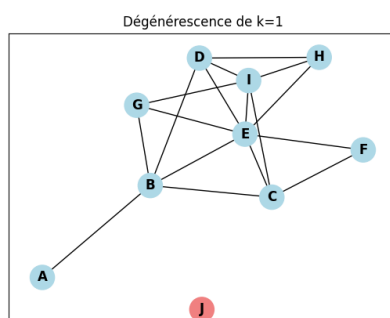


FIGURE 4 – Dégénérescence 1

ne sont pas inclus dans le sous-graphe. Ces sommets sont affichés en rouge pour indiquer qu'ils ne satisfont pas la contrainte de degré minimal égal ou supérieur à 2, conforme à la définition d'un 2-core.

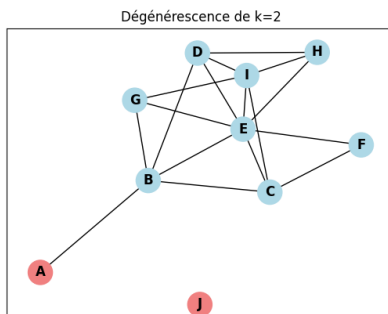


FIGURE 5 – Dégénérescence 2

Pour une dégénérescence dans laquelle $k = 3$, le graphe à la figure 6, ne regroupe pas dans le sous-graphe les sommets A , F et J . Pour cette dernière représentation, c'est le point F qui ne rentre pas dans les bonnes conditions.

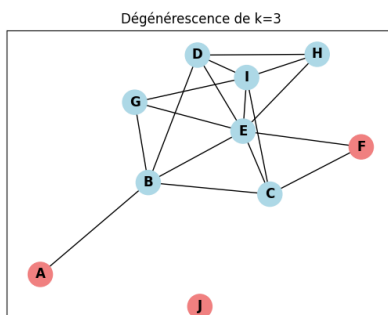


FIGURE 6 – Dégénérescence 3

La valeur maximale qui a été trouvée pour le graphe à la figure 6, est la valeur de $k = 3$, puisque si $k = 4$ alors tous les sommets ne feraient plus partie du sous-graphe, car aucun ne respectera la condition minimum de trois arêtes.

La dégénérescence d'un graphe offre une mesure de sa structure interne, en mettant en évidence des sous-ensembles rassemblant des sommets. Ces sommets forment un ensemble, ceux-ci sont tous connectés entre eux, pour former une clique. Cependant, malgré sa pertinence sur l'analyse des réseaux, la dégénérescence seule ne suffit pas toujours à capturer toutes les subtilités de la connectivité

dans un graphe. Cet ordre de dégénérescence sera exploité dans une variante d'un algorithme d'énumération de clique maximale qui seront expliquées dans les prochains chapitres.

Algorithme de Bron-Kerbosch

A présent, voici l'algorithme de Bron-Kerbosch réalisé en 1973 par Coenraad Bron et Joep Kerbosch [8], celui-ci a pour but d'énumérer les cliques maximales possibles dans un graphe G de façon récursive. Lors de son premier appel à l'algorithme, les paramètres R et X sont mis à 0, et P contient l'ensemble de tous les sommets du graphe G . R représente le résultat temporaire des sommets repris pour la future clique, P est l'ensemble des sommets candidats possible et X est l'ensemble des sommets qui sont exclus.

Plusieurs notations mathématiques sont importantes à comprendre pour la suite. Commençons par $\Gamma(R)$ qui est défini par l'ensemble des voisins de tous les sommets dans l'ensemble R . Pour illustrer notre propos, sur la figure 7, si on prend le point numéro 1, $\Gamma(R)$ vaut les valeurs 2, 3, 9. Ce sont bien les voisins pour le sommet numéro 1. Pour décortiquer un peu l'équation suivante, $P \cup X = \Gamma(R)$, $P \cup X$ représente l'union de l'ensemble P et de l'ensemble X , ce qui signifie que c'est un ensemble de tous les sommets qui sont soit candidats potentiels pour faire partie d'une clique, soit des sommets qui ont déjà été éliminés de la clique.

Pour continuer sur la bonne compréhension de l'algorithme, il est pertinent d'illustrer son comportement à travers un exemple concret. Pour ce faire, nous allons examiner un cas où l'algorithme sort les deux premières cliques maximales, voir 7 à 9.

Exemple Bron-Kerbosch

La figure 7 représente le graphique sur lequel va se dérouler notre exemple et sur lequel l'algorithme de Bron-Kerbosch va être exécuté.

Dans un premier temps, v prend la valeur de 1, c'est la première itération sur l'ensemble P . Celui-ci vaut $P = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, lors de la première exécution de l'algorithme, les différentes valeurs en paramètres sont $P \cap \Gamma(v) = \{9, 2, 3\}$, $R \cup \{v\} = \{1\}$ et $X \cap \Gamma(v) = \{\}$. La valeur de v passe à 9, ce qui fait que $P \cap \Gamma(v)$ et $X \cap \Gamma(v)$ sont tous deux vides, ce qui nous donne notre première clique maximale sur la figure 8. On rentre dans la condition de sortie de l'algorithme $P \cup X = \emptyset$, ce qui donne notre première clique maximale.

Ensuite, on passe à $v = 2$, ce qui implique que les valeurs de P et R changent

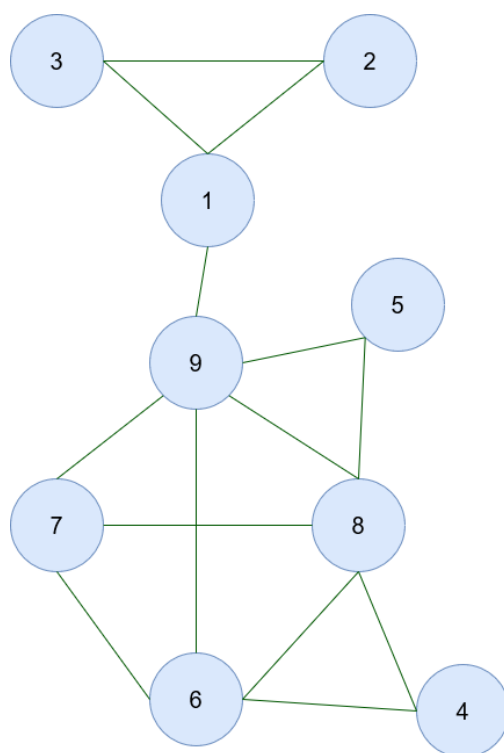


FIGURE 7 – Graphique Bron-Kerbosh

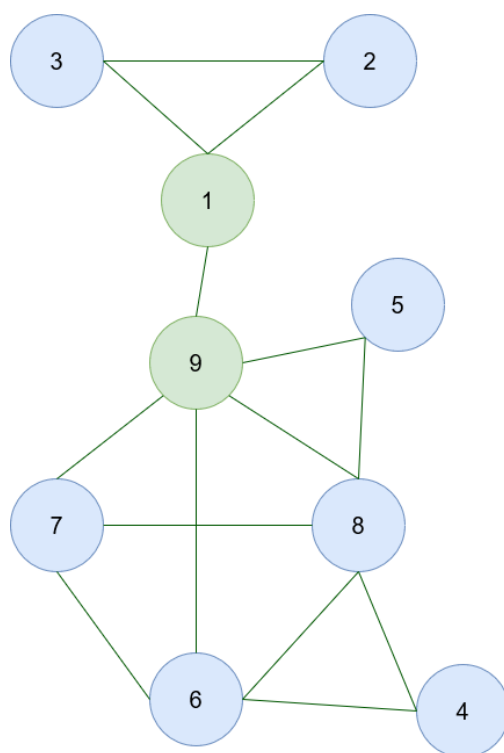


FIGURE 8 – Graphique Bron-Kerbosh 1-9

également, $P \cap \Gamma(v) = \{3\}$, $R \cup \{v\} = \{1, 2\}$ et $X \cap \Gamma(v) = \{\}$. On passe à la valeur $v = 3$, ce qui a pour effet de rentrer dans la condition de sortie pour lesquelles P et X sont vides, ce qui donne notre deuxième clique maximale à la figure 9.

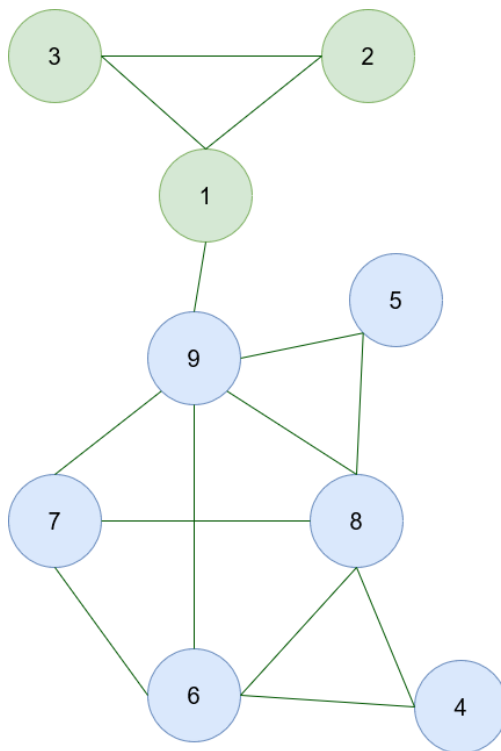


FIGURE 9 – Graphique Bron-Kerbosh 1-2-3

Au final, l'algorithme détecte six cliques maximales, celles-ci sont les deux premières pour lesquelles les étapes ont été démontrées, ainsi que les cliques : $R = \{8, 4, 6\}$, $R = \{8, 9, 5\}$, $R = \{8, 9, 6, 7\}$, $R = \{9, 6, 7\}$. La constatation qui est faite est que l'algorithme de base de Bron-Kerbosh ne gère pas les doublons, c'est pour cette raison que nous avons les cliques $R = \{8, 9, 6, 7\}$ et $R = \{9, 6, 7\}$.

À présent, que nous avons vu pas à pas le fonctionnement pour trois cliques maximale, penchons-nous sur le fonctionnement de l'algorithme de Bron-Kerbosch. C'est ce que va traiter le prochain chapitre de notre article.

Fonctionnement Bron-Kerbosch

Lors de l'explication du fonctionnement de l'algorithme, le but bien précis de ce dernier est d'énumérer les cliques maximales présentes dans un graphe G . Pour ce faire, celui-ci se déroule en plusieurs étapes. Au début de l'algorithme,

une condition de sortie est prévue pour savoir si l'ensemble P et X sont vides. Si ceux-ci le sont, plus aucun sommet candidat ne peut être ajouté à la clique. Celle-ci devient donc une clique maximale, on retourne la valeur de l'ensemble de R . On itère ainsi à travers tous les sommets de $P(v)$, en les ajoutant un par un à R et en les retirant de P et X . A l'intérieur de cette itération, on utilise à nouveau la fonction de Bron-Kerbosh, on passe en premier paramètre l'ensemble de tous les sommets de P qui sont voisins de v ($P \cap \Gamma(v)$). Le deuxième paramètre, est la clique qui est en cours de construction avec le sommet v , noté $R \cup \{v\}$. Ensuite en troisième paramètre, les sommets de X qui sont voisins de v , avec la notation suivante : $X \cap \Gamma(v)$. Ce qui suit l'appel à la fonction est la suppression du sommet v de l'ensemble P et l'ajout du somme v à l'ensemble X . L'algorithme continue son itération jusqu'à avoir parcouru tous les sommets de l'ensemble P .

Ceci est le pseudo-code de l'algorithme de Bron Kerbosch présenté dans l'article "Listing all maximal cliques in sparse graphs in near-optimal time" [3].

Function BronKerbosch(P, R, X)

- 1: **if** $P \cup X = \emptyset$ **then**
- 2: report R as a maximal clique
- 3: **end if**
- 4: **for** each vertex $v \in P$ **do**
- 5: BronKerbosch($P \cap \Gamma(v), R \cup \{v\}, X \cap \Gamma(v)$)
- 6: $P \leftarrow P \setminus \{v\}$
- 7: $X \leftarrow X \cup \{v\}$
- 8: **end for**

La modélisation suivante permet de visualiser l'algorithme d'une autre manière avec le détail de chaque paramètre à chaque itération de l'algorithme 10.

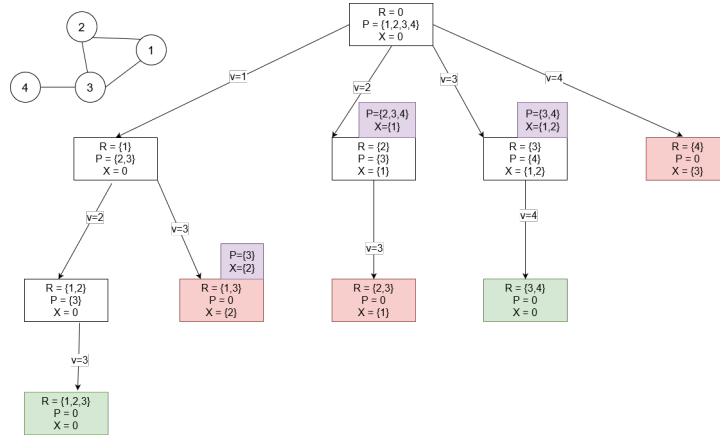


FIGURE 10 – Diagramme Bron-Kerbosch

Le prochain chapitre traite d'une variante de l'algorithme avec l'ajout d'un point de pivot. Celui-ci a pour effet de peut-être réduire les appels récursifs, la suite et les explications, nous donnerons la réponse à cette question.

Bron-Kerbosch Tomita et al

Dans ce chapitre, l'algorithme de Bron-Kerbosch est un peu modifié avec l'ajout d'un point de pivot. Celui-ci a pour objectif de réduire le nombre d'appels récurifs sur ce dernier. La notion clé est que pour tout sommet u dans $P \cup X$, que l'on nommera pivot, toute clique maximale contient l'un des non-voisins de u . Ceci implique le fait de retarder l'ajout de sommet dans $P \cap \Gamma(u)$ à la clique. Dans lequel $\Gamma(u)$ représente les voisins de u . Tomita et al. [7] garantit que l'algorithme de Bron-Kerbosch a un temps d'exécution dans le pire des cas de $O(3^{n/3})$. Les détails de la complexité de l'algorithme n'est pas détaillé dans l'article.

Prenons un exemple avec un graphe G de sommets $\{1, 2, 3, 4\}$, lors de l'initialisation, P vaut $\{1, 2, 3, 4\}$ et R et X sont initialisés à vide. À présent, on choisit le point de pivot u dans $P \cup X$, dans notre exemple 11, le sommet 3 est la valeur qui maximise $|P \cap \Gamma(3)|$. Les voisins de $\Gamma(3)$ est l'ensemble $\{1, 2, 4\}$ et les non-voisins dans P sont $P \setminus \Gamma(3) = \{\}$.

Lors de l'itération de la boucle à la ligne n°5 de l'algorithme de BronKerboschPivot, on ne choisit pas librement la valeur v dans P , on choisit v dans P sans les voisins du pivot. On exclu les voisins du pivot, ce qui va limiter le nombre d'itérations. Ici dans notre exemple, $P \setminus \Gamma(3)$ est vide, on ne fait aucun appel récurif pour cette étape, on passe à la deuxième itération avec la mise à jour des ensembles P et X . On retire 3 de P et on l'ajoute à X . Les valeurs pour P sont l'ensemble $\{1, 2, 4\}$ et X vaut 3. On choisit un nouveau pivot parmi l'ensemble $\{1, 2, 3, 4\}$, on suppose $u = 2$, on calcule $P \setminus \Gamma(2)$, pour les valeurs $\Gamma(2) = \{1, 3\}$ et $P \setminus \Gamma(2) = 4$. Le schéma 11 montre bien le détail pas à pas de l'algorithme sur le graphe G . On obtient à la fin deux cliques maximales, la clique $\{1, 2, 3\}$ et la clique $\{3, 4\}$.

Le pseudo-code de l'algorithme Bron-Kerbosch avec un point de pivot vient de l'article [3]. Les auteurs ont prouvé grâce à cet algorithme la génération de toutes les cliques maximales sans duplication.

Function BronKerboschPivot(P, R, X)

- 1: **if** $P \cup X = \emptyset$ **then**
- 2: report R as a maximal clique

```

3: end if
4: choose a pivot  $u \in P \cup X$   $\triangleright$  Tomita et al. choose  $u$  to maximize  $|P \cap \Gamma(u)|$ 
5: for each vertex  $v \in P \setminus \Gamma(u)$  do
6:   BRONKERBOSCHPIVOT( $P \cap \Gamma(v)$ ,  $R \cup \{v\}$ ,  $X \cap \Gamma(v)$ )
7:    $P \leftarrow P \setminus \{v\}$ 
8:    $X \leftarrow X \cup \{v\}$ 
9: end for

```

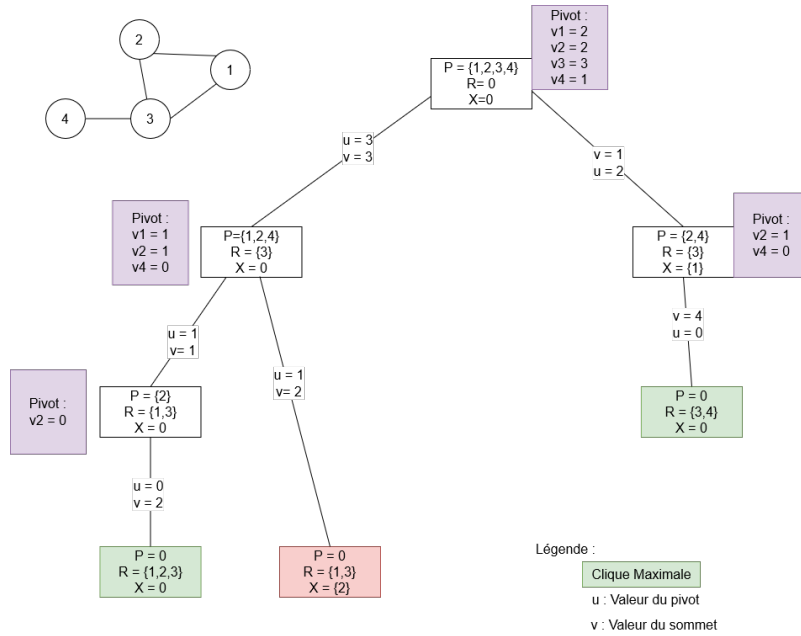


FIGURE 11 – Diagramme Bron-Kerbosch Pivot

Le prochain chapitre va utiliser une notion vue précédemment ainsi que l'utilisation d'un point pivot.

Bron-Kerbosch et la Dégénérescence

Ce chapitre est consacré à l'explication de l'algorithme proposé par les auteurs dans leur article [3]. Ils y présentent une amélioration de l'algorithme de Bron-Kerbosch en exploitant la dégénérescence, abordée dans la section 2, ainsi que la stratégie de pivot.

Pour une bonne compréhension du fonctionnement de l'algorithme, la figure 12 permettra de suivre pas à pas les valeurs obtenues. La première observation faite par les auteurs [3] est que lors du traitement d'une clique R , les voisins communs de R peuvent être partitionnés en deux ensembles : P , qui comprend les sommets situés après le dernier sommet de R , et X , qui contient les voisins restants.

L'utilisation de la stratégie de pivot de Tomita et al. [7] permet de générer les cliques maximales qui contiennent chaque sommet dans R , quelques sommets dans P et aucun sommet dans X , sans doublons. (Voir la preuve de Tomita et al. [7])

Concernant la dégénérescence, soit C une clique maximale, et v son premier sommet dans l'ordre de dégénérescence. D'après Tomita et al [7], C est reportée une seule fois lors du traitement de v . Si l'on traite un autre sommet de C , v est placé dans X . Dans notre exemple, l'ordre de dégénérescence est le suivant : $\{4, 1, 2, 3\}$. Le sommet 4 a comme voisin le sommet 3, ce qui en fait le premier dans l'ordre, tandis que le sommet 3 a 3 sommets ($\{1, 2, 4\}$) comme voisins, ce qui en fait le dernier.

Lors de l'appel à `BronKerboschPivot`, on passe un sous-graphe $H(P, X)$. Au début, $P = 3$ et $X = 0$. On obtient alors $R = 4$, ce qui nous donne finalement une clique maximale $R = \{3, 4\}$. Celle-ci répond à la règle qu'une arête (u, v) de G est sélectionnée comme une arête de $H(P, X)$ si le pivot u ou le sommet v appartient à P et que les deux appartiennent à l'ensemble $P \cup X$. Le pivot choisi selon Tomita et al. [7] est celui qui a le plus de voisins dans P .

Ensuite, l'algorithme prend le sommet 1, suivant l'ordre de dégénérescence. Cet ordre est calculé en retirant de façon itérative le sommet ayant le moins d'arêtes

avec ses voisins. Les sommets 1 et 2 sont les suivants. On obtient les valeurs $\{2, 3\}$ pour P , $R = 1$ et $X = 0$. Ensuite, le sommet 2 est sélectionné, les valeurs de l'ensemble R sont $\{1, 2\}$, $P = 3$ et $X = 0$. La dernière itération sur cet ensemble donne la dernière clique maximale $R = \{1, 2, 3\}$.

La principale différence se trouve à la ligne n°1 de l'algorithme de BronKerbosch-Degeneracy, où l'on utilise cette notion de dégénérescence pour favoriser un ordre de passage pour BronKerboschPivot à la ligne 4. Grâce à cet ordonnancement, les auteurs ont démontré qu'ils ont pu obtenir un temps de $O(dn3^{d/3})$. La variable d représente le plus petit entier tel que chaque sous-graphe du graphe contient un sommet avec au plus d voisins.

L'algorithme de BronKerboschDegeneracy vient de l'article des auteurs [3]. Les deux paramètres de l'algorithme sont l'ensemble des sommets du graphe, noté V , et l'ensemble des arêtes du graphe, noté E .

Function BronKerboschDegeneracy(V, E)

```

1: for each vertex  $v_i$  in a degeneracy ordering  $v_0, v_1, v_2, \dots$  of  $(V, E)$  do
2:    $P \leftarrow \Gamma(v_i) \cap \{v_{i+1}, \dots, v_{n-1}\}$ 
3:    $X \leftarrow \Gamma(v_i) \cap \{v_0, \dots, v_{i-1}\}$ 
4:   BRONKERBOSCHPIVOT( $P, \{v_i\}, X$ )
5: end for
```

Pour vérifier la différence entre l'algorithme avec pivot et celui avec dégénérescence, une petite implémentation en python est réalisée affichant le temps d'exécution et le nombre d'itérations. Les valeurs obtenues sont faites avec l'exemple présent pour ces deux algorithmes, celui de BronKerboschPivot obtient $NbIterations = 5$ et $Temps = 0.00066$ secondes, BronKerboschDegeneracy obtient $NbIterations = 7$ et $Temps = 0.00018$ secondes. On remarque une différence au niveau du nombre d'exécutions entre les deux, celle-ci peut s'expliquer par le fait que celui avec dégénérescence explore chaque branche du graphe, même si elle semble redondante, pour s'assurer que toutes les cliques maximales sont découvertes. Notre graphe est très petit, ce qui ne permet pas forcément de constater l'amélioration entre celui avec pivot et celui avec la dégénérescence, même si au niveau du temps d'exécution, celui avec dégénérescence est plus rapide. Celui avec pivot peut varier et conduire à un nombre d'itérations moindre si le pivot choisi est très efficace pour réduire rapidement P . L'algorithme BronKerboschPivot peut limiter le nombre de branches à explorer en excluant les voisins du pivot, tandis que celui avec dégénérescence explore chaque possibilité. En revanche, ce sont sur des graphes plus grands ou plus complexes, que l'algorithme de BronKerboschDegeneracy montre une efficacité supérieur en terme de réduction des branches explorées.

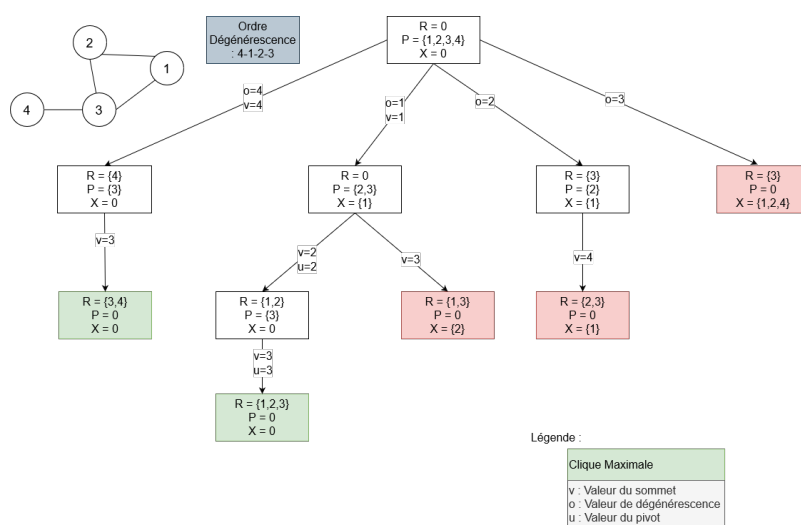


FIGURE 12 – Diagramme Bron-Kerbosch Dégénérescence

Conclusion

Au cours de ce travail, nous avons exploré en profondeur les algorithmes de recherche de cliques maximales dans les graphes, en mettant l'accent sur l'algorithme de Bron-Kerbosch et ses améliorations. Nous avons commencé par poser les bases nécessaires à la compréhension de notre étude en définissant les concepts et le vocabulaire essentiels en théorie des graphes.

Le deuxième chapitre nous a permis d'examiner l'algorithme de Bron-Kerbosch dans sa forme originale, fournissant une compréhension détaillée de ses mécanismes et de ses performances. Nous avons ensuite introduit une optimisation clé dans le troisième chapitre : l'ajout d'un point de pivot selon l'approche de Tomita et al., démontrant comment cette modification améliore l'efficacité de l'algorithme.

Enfin, le dernier chapitre a exploré l'algorithme de BronKerboschDegeneracy, qui utilise la notion de dégénérescence des graphes et la combinaison avec le point de pivot pour optimiser la recherche de cliques maximales. Nous avons montré que le problème est traitable par des paramètres fixes en terme de dégénérescence du graphe. Cette légère modification permet d'atteindre des performances optimales en fonction de la dégénérescence.

Pour conclure, la dégénérescence est un atout qui permet de diminuer le temps d'exécution de l'algorithme suite au fait que l'exposant est sur d divisé par 3, comparé à l'algorithme de base qui a une complexité en exposant n . Ce qui signifie qu'un graphe qui a une faible dégénérescence va pouvoir déterminer les cliques d'une manière beaucoup plus rapide comparé à un graphe qui a une dégénérescence plus élevée ou si sur ce graphe on utilise l'algorithme de base.

Bibliographie

- [1] Immanuel M BOMZE. “Evolution towards the maximum clique”. In : *Journal of Global Optimization* 10 (1997), p. 143-164.
- [2] Austin BUCHANAN et al. “Solving maximum clique in sparse graphs : an $O(nm + n^2d/4)$ algorithm for d -degenerate graphs”. In : *Optimization Letters* 7.7 (2013). Original Paper, p. 1513-1523. DOI : 10.1007/s11590-013-0698-2.
- [3] David EPPSTEIN, Maarten LÖFFLER et Darren STRASH. “Listing all maximal cliques in sparse graphs in near-optimal time”. In : *Algorithms and Computation : 21st International Symposium, ISAAC 2010, Jeju Island, Korea, December 15-17, 2010, Proceedings, Part I* 21. Springer. 2010, p. 403-414.
- [4] Yi-Xiu KONG et al. “k-core : Theories and applications”. In : *Physics Reports* 832 (2019), p. 1-32.
- [5] Edmund R. PEAY. “Hierarchical Clique Structures”. In : *Sociometry* 37.1 (1974), p. 54-65.
- [6] Stephen B SEIDMAN. “Network structure and minimum degree”. In : *Social networks* 5.3 (1983), p. 269-287.
- [7] Etsuji TOMITA, Akira TANAKA et Haruhisa TAKAHASHI. “The worst-case time complexity for generating all maximal cliques and computational experiments”. In : *Theoretical computer science* 363.1 (2006), p. 28-42.
- [8] WIKIPÉDIA. “Algorithme de Bron-Kerbosch”. In : (3 mars 2024). Consulté le 07 Mai 2024. URL : https://fr.wikipedia.org/wiki/Algorithme_de_Bron-Kerbosch.
- [9] WIKIPÉDIA. “Degré (théorie des graphes)”. In : (20 septembre 2023). Consulté le 03 Février 2024. URL : [https://fr.wikipedia.org/wiki/Degr%C3%A9_\(th%C3%A9orie_des_graphes\)](https://fr.wikipedia.org/wiki/Degr%C3%A9_(th%C3%A9orie_des_graphes)).
- [10] WIKIPÉDIA. “Sous-graphe”. In : (30 août 2023). Consulté le 04 Mai 2024. URL : <https://fr.wikipedia.org/wiki/Sous-graphe>.