UVSQ université PARIS-SACLAY
ISTY
Institut des Sciences et Techniques des Yvelines
CAMPUS DE MANTES EN YVELINES
CAMPUS DE SAINT-QUENTIN-EN-YVELINES

Recherche de Cliques Maximales dans des Graphes Creux

PROJET DE PROGRAMMATION NUMÉRIQUE

Achrrab Rim
Bouamara Nedjma
Chikhi Katia

Encadré par :
M. MANOUSSAKIS George

CALCUL HAUTE PERFORMANCE SIMULATION

Résumé

Le problème de l'énumération de cliques maximales MCE implique la recherche de toutes les cliques maximales d'un graphe non orienté donné. C'est un problème NP-difficile, dont la difficulté de trouver les cliques maximales augmente en fonction de la taille du graphe étudié.

Notre projet consiste à utiliser l'algorithme de Bron Kerbosch pour résoudre le problème d'énumération de cliques maximales en utilisant un paramètre important du graphe qui est la dégénérescence, cette approche permet d'améliorer la complexité.

Mots-clés : - MCE, Clique,Clique maximale, Algorithme de Bron Kerbosch.

Abstract

The MCE maximal clique enumeration problem involves finding all maximal cliques of a given undirected graph. It is an NP-hard problem, the difficulty of finding the maximal cliques increases with the size of the studied graph.

Our project consists in using the Bron Kerbosch algorithm to solve the maximum cliques enumeration problem by using an important parameter of the graph which is the degeneracy, this approach allows us to improve the algorithm's complexity.

Keywords : - MCE, Clique, Maximum clique, Bron Kerbosch algorithm.

Table des matières

Table des figures

Liste des tableaux

Nomenclature

MCE **M**aximal **C**lique **E**numeration

MV **M**achine **V**irtuelle

AVX **A**dvanced **V**ector **E**xtensions

RAM **R**andom **A**ccess **M**emory

SMP **S**ymmetric **M**ulti-**P**rocessing

CHAPITRE 1

Introduction générale

Les graphes non orientés sont des structures de données qui nous permettent de modéliser des objets ou des problèmes avec de nombreuses applications, notamment dans les domaines des réseaux sociaux, des réseaux de transport ou encore de la biologie. Parmi ces applications, on trouve la recherche de structures, telles que la recherche de communautés dans les graphes issus des réseaux sociaux.

Dans notre projet, nous nous intéresserons à la recherche de cliques maximales dans des graphes creux. Cela consiste à lister toutes les cliques maximales d'un graphe non orienté, en utilisant l'algorithme de Bron Kerbosch avec une nouvelle méthode de décomposition basée sur la dégénérescence d'un graphe, on va implémenter en premier lieu cet algorithme puis on va le paralléliser et voir la différence en terme de performances.

Première partie

partie séquentielle

CHAPITRE 2

Contexte général du projet

2.1 Introduction

Dans ce chapitre, nous allons élaborer le cadre général de notre projet en spécifiant la problématique à résoudre,les objectifs, les outils utilisés et les éléments fondamentaux de notre étude.

2.2 Définitions

Graphe

Un graphe est une structure de données définie par un couple G = (S, A) tel que :

— S est un ensemble fini de sommets (ou nœuds), un sommet est tout simplement n'importe quelle entité représentée par les données (par exemple : un utilisateur de réseaux sociaux).

— A est un ensemble d'arêtes ou des relations reliant ces sommets, tel que :
$A \subseteq \{(x, y)|x, y \in S^2 \wedge x \neq y\}$.

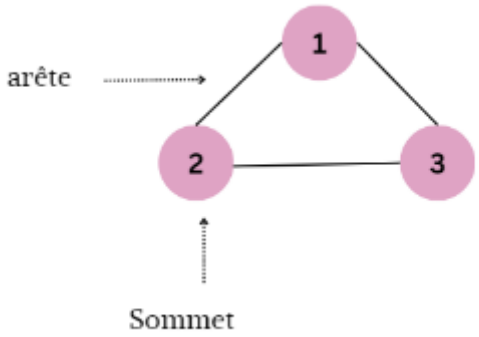


FIGURE 2.1 – Sommets et arêtes d'un graphe

Graphe non orienté

Un graphe est dit non orienté si les arêtes n'ont pas de sens entre deux sommets : si un sommet s_1 est relié à un autre sommet s_2 par un arc, le sommet s_2 est également relié à s_1.

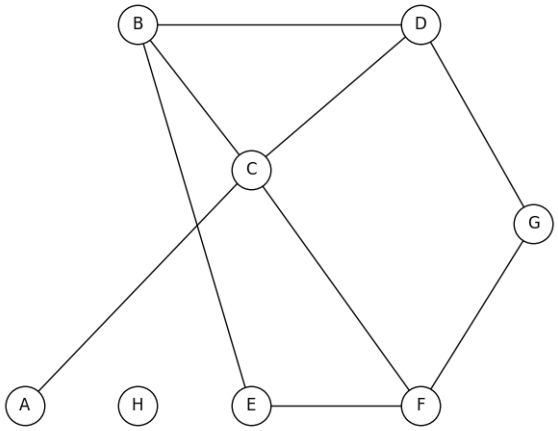


FIGURE 2.2 – graphe non orienté

Graphe complet

un graphe G est dit complet s'il comporte une arête (s_i, s_j) pour toute paire de sommets différents $(s_i\ ,\ s_j) \in S$ (S est l'ensemble des sommets contenants dans le graphe G).

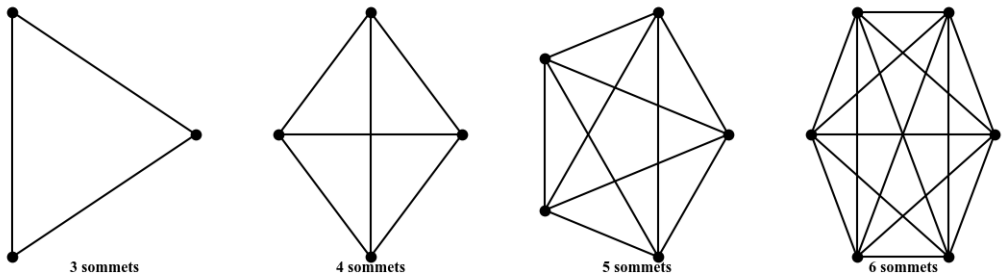


FIGURE 2.3 – Graphes complets

Ordre d'un graphe

L'ordre d'un graphe est défini comme le nombre fini de sommets qu'il contient.
Si par exemple un graphe possède six sommets, on dit qu'il est d'ordre six.

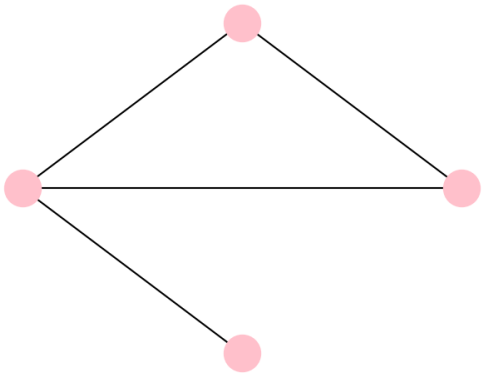


FIGURE 2.4 – Graphe d'ordre 4

Sous-graphe

Soit G=(S,A) un graphe qui contient un ensemble fini S de sommets et A d'arrêts. On dit que $G^{'} = (S^{'}, A^{'})$ est un sous-graphe de G si et seulement si :

— S $\subseteq S^{'}$

— A $\subseteq A^{'}$

Dans la figure ci-dessous, le graphe $\{A, B, C, D\}$ est un sous-graphe du graphe G.

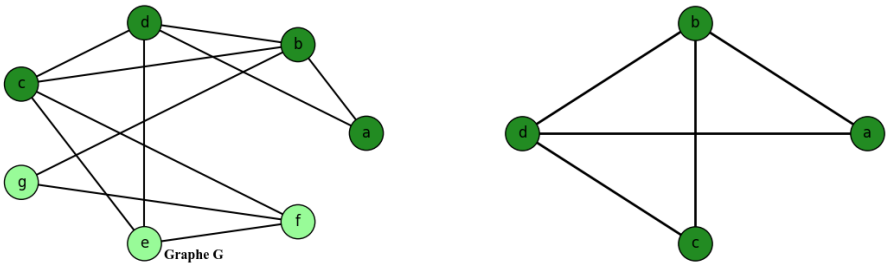


FIGURE 2.5 – Sous-graphe

Sous-graphe induit

Étant donné un graphe non orienté G=(S,A) d'ordre fini n. Si on supprime des sommets s_i , $i \in (0, 1, ..., n)$ en éliminant tous les arcs reliés à ces sommets, on obtient un sous-graphe induit de G.

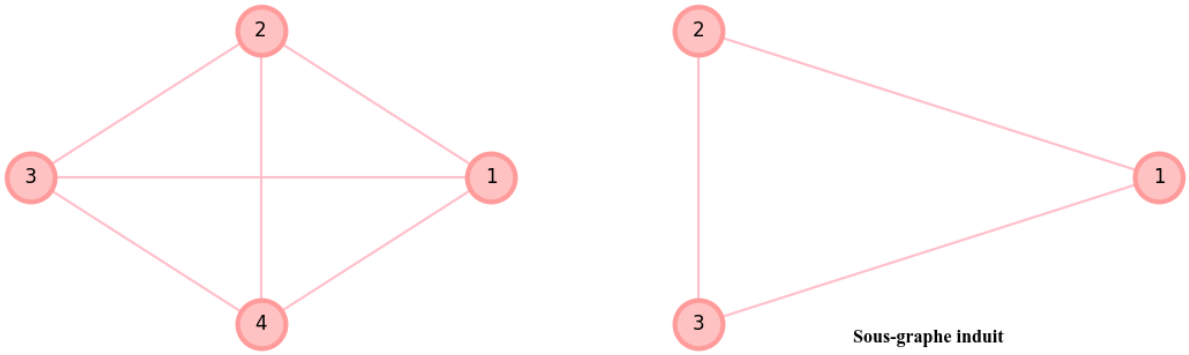


FIGURE 2.6 – Sous-graphe induit

Clique d'un graphe

Une clique dans un graphe G est un sous-ensemble de sommets de G tel que chaque sommet est relié à tous les autres par des arêtes (ou arcs), une clique est donc un sous-graphe complet d'un graphe donné G.

Dans l'exemple ci-dessous, on remarque que les sommets $\{1, 2, 3\}$ forment une clique.

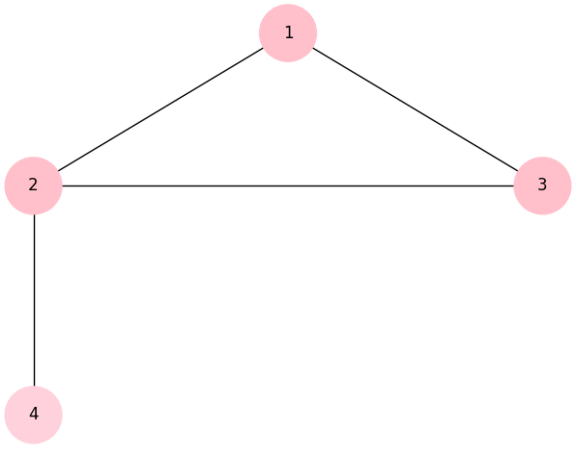


FIGURE 2.7 – Clique d'un graphe

Clique maximale

Une clique est dite maximale si elle n'est pas incluse dans une clique de taille supérieure. c'est-à-dire qu'elle ne peut pas être étendue en ajoutant un sommet supplémentaire.

Dans l'exemple ci-dessous :

— $\{2, 3, 4, 5\}$ est une clique maximale

— $\{2, 3, 5\}$ n'est pas une clique maximale car elle est incluse dans l'ensemble des sommets $\{2, 3, 4, 5\}$.

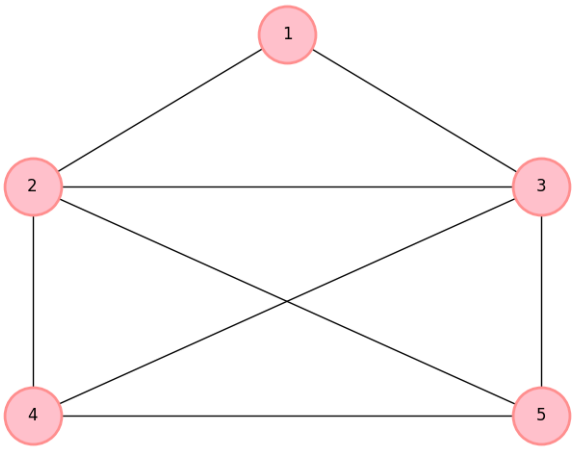


FIGURE 2.8 – Clique maximale dans un graphe

Degré d'un sommet

Soit G=(S,A) un graphe non orienté fini d'ordre n. Le degré d'un sommet s_i , $i \in 1, 2, ..., n$ avec $n \in N$, que l'on note $d(s_i)$ est le nombre d'arêtes incidentes à ce sommet. Dans le graphe ci-dessous, les degrés des sommets $\{0, 1, 2, 3\}$ sont les suivants :
d(0)=d(3)=3, d(1)=2, et d(2)=2.

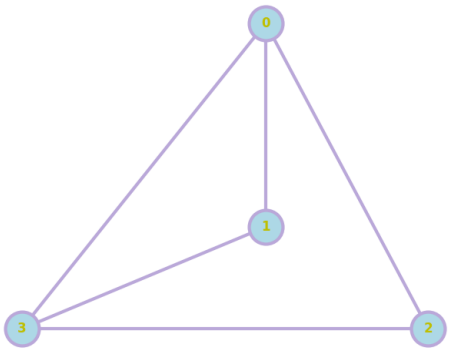


FIGURE 2.9 – Degré des sommets d'un graphe

Graphe creux

Un graphe creux (sparse graph en anglais) est un graphe dont ses sommets ont des degrés faibles (identiquement équivalent les sommets ne sont pas connectés à plusieurs voisins).

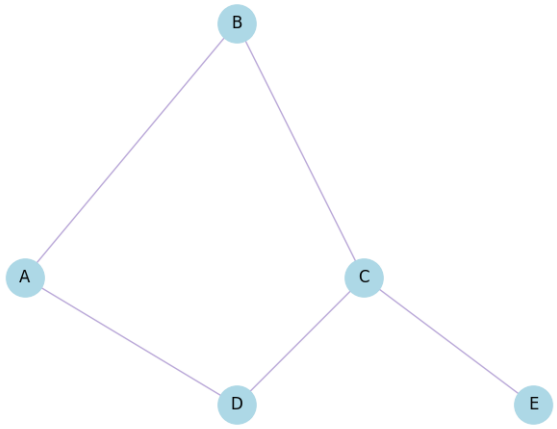
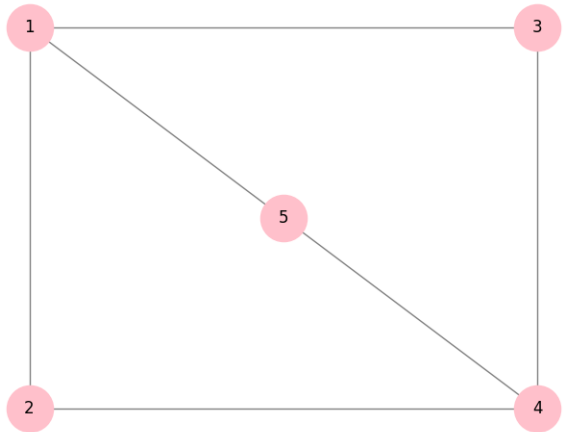


FIGURE 2.10 – Graphe creux

Dégénérescence d'un graphe

La dégénérescence d'un graphe est une mesure qui est associée à un graphe non orienté, elle permet de mesurer à quel point un graphe est creux. On dit qu'un graphe G est k-dégénéré si pour tout sous-graphe H de G, H contient au moins un nœud de degré inférieur ou égal à k, La dégénérescence de G est le plus petit nombre entier k tel que G est k-dégénéré.

Le graphe ci-dessous est un graphe 2-dégénéré



L'ordre de dégénérescence, noté σ_G est un ordre sur les sommets qu'on obtient en supprimant à chaque fois le sommet qui a le plus petit degré dans le sous-graphe restant. L'ordre de dégénérescence dans le graphe ci-dessous est $['4','1','2','3']$.

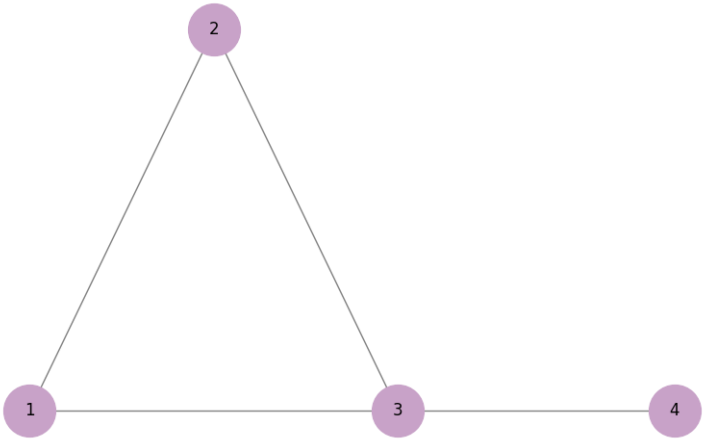


FIGURE 2.11 – Ordre de dégénéréscence d'un graphe

Table de hashage

Une table de hachage est une structure de données qui permet une association clé–valeur. En utilisant une table de hachage, nous sommes en mesure de définir chaque sommet particulier comme une clé de hachage, et son appariement de valeur à un tableau de sommets adjacents.La raison de l'utilisation d'une table de hachage est la recherche rapide des liens entre les sommets dans les graphes qui fait (O(1)) en temps.

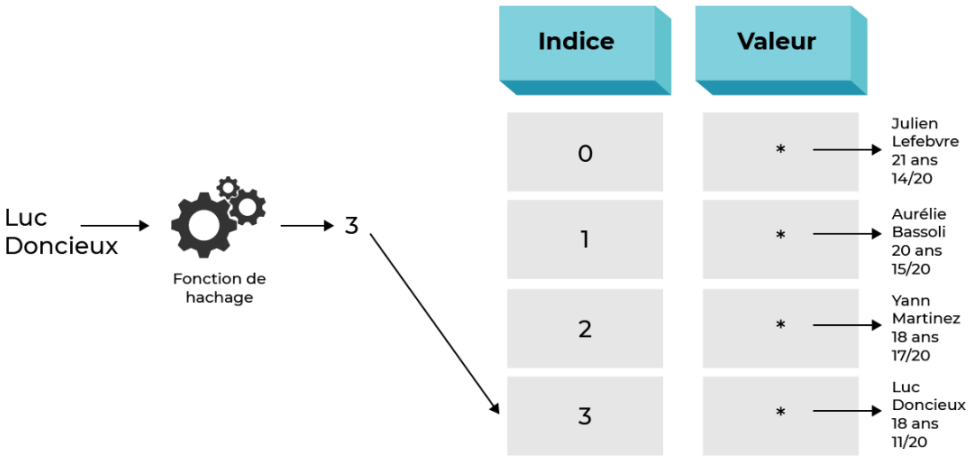


FIGURE 2.12 – Un exemple de table de hashage

Recherche de cliques maximale

Pour rechercher les cliques maximales dans des structures de données, plusieurs algorithmes existent pour résoudre le problème MCE. L'algorithme de Bron Kerbosh est un algorithme d'énumération qui a été créé en 1973 pour trouver toutes les cliques maximales dans un graphe non orienté.

Il peut être utilisé pour détecter les communautés étroites dans des réseaux sociaux comme Facebook en traitant chaque personne comme un nœud et chaque lien d'amitié comme un arc reliant les deux personnes dans le graphe.

Avant d'appliquer l'algorithme de Bron Kerbosh, il faut d'abord créer la représentation graphique qui convient le réseau social.Une fois que vous avez la représentation graphique, vous pouvez ensuite appliquer l'algorithme de Bron-Kerbosch pour trouver les cliques dans le graphique.

L'algorithme de Bron-Kerbosch est un moyen efficace pour détecter les communautés étroites dans les grands réseaux sociaux, car il a une complexité temporelle de $O(3^{(n/3)})$ et $O(n^2)$ en mémoire, où n est l'ordre du graphe ($n \in \mathbb{N}$). Cependant, il convient de noter que l'algorithme ne peut détecter que les cliques et peut ne pas être en mesure d'identifier des types plus généraux des communautés étroites.

2.3 Description des données

Le jeu de données représente un graphe qui a été généré à partir des données de courrier électronique d'une grande institution de recherche européenne. Les nœuds représentent les membres de l'institution. Il existe une arête (u, v) dans le graphe, si la personne u a envoyé au moins un e-mail à la personne v. Le graphe donné est orienté, on l'a transformé en un graphe non orienté pour qu'on puisse appliquer l'algorithme de cliques maximales. Le tableau suivant décrit l'ensemble de données :

Statistique de jeu de données	
Noeuds	1005
Bords	25571
Noeuds dans le plus grand WCC	986 (0.981)
Bords dans le plus grand WCC	25552 (0.999)
Noeuds dans le plus grand SCC	803 (0.799)
Bords dans le plus grand SCC	24729 (0.967)
Coefficient de regroupement moyen	0.3994
Nombre de traingles	105461
Fraction de triangles fermés	0.1085
Diamètre (chemin le plus long et le plus court	7
'Diamètre effectif à 90 centiles	2.9

TABLE 2.1 – Description de l'ensemble des données

2.4 Problématique

L'objectif de notre travail est d'implémenter un algorithme optimal pour résoudre le problème de clique MCE. Ce projet vise à répondre aux défis suivants :

Problème 1 : Trouver manuellement toutes les cliques maximales dans les grands graphes tels que les réseaux de transport ou les réseaux sociaux est une tâche difficile voir impossible.

Problème 2 : Les algorithmes existants pour trouver les cliques maximales ne sont pas optimisés, ce qui pose des problèmes de complexité en temps et en mémoire. Ceci est particulièrement problématique dans les graphes de grande taille, ce qui rend le problème encore plus complexe.

Objectifs

Après avoir détecter les problèmes, voilà nos objectifs à atteindre :

— L'objectif principal de ce projet consiste à modéliser une solution informatique qui va mettre en présence un algorithme en C++ qui va détecter toutes les cliques maximales contenues dans un graphe non orienté.

— La mise en place d'une structure de données idoine pour la représentation graphique afin de faciliter l'accès aux informations désirées.

— Réduire l'intervalle de temps de l'exécution de l'algorithme pour qu'on puisse rapidement trouver les sous-graphes complets du graphe principal.

2.5 Environnement de travail

Environnement matériel (Hardware)

Nous avons utilisé pour ce projet trois ordinateurs ayant les caractéristiques suivantes :

Ordinateur portable : HP EliteBook 840 G5

— Système d'exploitation : Windows 10 Professionnel 64 bits

— Processeur : intel(R) Core(TM) i5-7300U CPU @ 2.60 GHZ

— Mémoire RAM : 8GB

— Carte mère : HP 83B2 KBC version 23.53.00

Ordinateur portable : HP laptop 15s-fq2xxx 11th Gen

— Système d'exploitation : Windows 11 Professionnel 64 bits

— Processeur : Intel(R) Core(TM) i3-1125G4 @ 2.00GHz

— Mémoire RAM : 8GB

— Espace disque : 256 GB SSD

Ordinateur portable : Hewelett-Packard HP 250 G2 Notebook PC

— Système d'exploitation : Ubuntu 22.04.1 LTS 64 bits

— Processeur : Intel(R) Core(TM) i3-3110M CPU @ 2.40GHz x4

— Mémoire RAM : 6GB

— Carte graphique : Mesa Intel(R) HD Graphics 4000 (IVB GT2)

— Espace disque : 120 GB

Machine virtuelle

FIGURE 2.13 – Logo d'une MV

C'est un environnement virtuel indépendant du système d'exploitation de la machine physique, il contient son propre système, sa propre mémoire, son propre espace et ses propres paramètres de cofiguration. Nos ordinnateurs ne disposent pas du système d'exploitation Unix, donc on aura besoin d'une machine virtuelle pour installer Ubuntu.

Linux

Linux est un système d'exploitation gratuit basé sur Unix.

Il a été créé en 1991 par Linus Torvalds. Il est conçu pour être utilisé dans une variété d'appareils et serveurs, superordinateurs, ordinateurs de bureau, smartphones et les appareils intégrés.

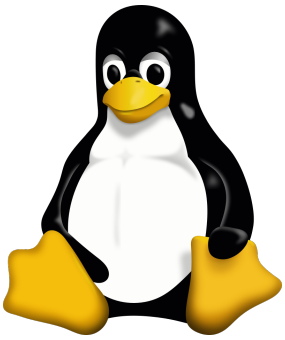


FIGURE 2.14 – Logo Linux

Latex

FIGURE 2.15 – Logo Latex

LaTeX est un système de traitement de texte et préparation de documents pour une composition de haute qualité. Il est le plus souvent utilisé pour des documents techniques ou scientifiques de différentes tailles et thèmes.

C++

C'est un langage de programmation compilé de haut niveau qui prend en charge plusieurs paradigmes de programmation, y compris la programmation procédurale,orientée objet et fonctionnelle. Il permet de faire la gestion de la mémoire, la surcharge d'opérateurs et de fonctions, la manipulation des exceptions.

FIGURE 2.16 – Logo C++

Python

FIGURE 2.17 – Logo Python

Python est un langage de programmation interprété de haut niveau, créé par Guido van Rossum en 1991. Il est largement utilisé par la communauté des développeurs en raison de la productivité accrue qu'il offre. Il prend en charge plusieurs paradigmes de programmation, y compris la programmation structurée, orientée objet et fonctionnelle. Python est utilisé dans tous les domaines, de l'apprentissage automatique à la création de sites Web, en passant par le test de logiciels. Il est multiplateforme et ses programmes peuvent s'exécuter sous Windows, Linux et macOS.

Numpy

Elle Signifie Python numérique. C'est une bibliothèque Python open source à usage général qui fournit des outils pour gérer les tableaux à n dimensions. NumPy offre à la fois la flexibilité de Python et la vitesse d'un code C compilé bien optimisé. Sa syntaxe facile à utiliser le rend hautement accessible et productif pour les programmeurs de tous horizons.

FIGURE 2.18 – Logo Numpy

Matplotlib

FIGURE 2.19 – Logo Matplotlib

C'est une bibliothèque open source ,utilisée pour créer des visualisations statiques, et interactives en Python.Elle a été créée par John D. Hunter en 2002, principalement écrite en python avec quelques segments écrits en C, Objective-C et Javascript. Elle offre des tracés de qualité, des figures interactives qui peuvent zoomer, faire un panoramique, mettre à jour, en plus de l'utilisation d'un large éventail de packages tiers construits sur Matplotlib. Cette bibliothèque offre une flexibilité accrue, c'est la meilleure option lorsque les performances sont parfois supérieures.

Outils et frameworks

Visual Studio Code

Lancé en 2015 par le géant Microsoft, visual studio est un éditeur du code source développé par Microsoft pour Windows, Linus et MacOs, avec prise en charge intégrée de plusieurs langages de programmation, et il fournit plusieurs extensions pour autres langages (tels que C++, C#, Java, Python) qui facilite aux développeurs l'écriture du code et le débogage. Visual Studio inclut des compilateurs, des outils de complétion de code,et bien d'autres fonctionnalités pour améliorer le processus du développement.

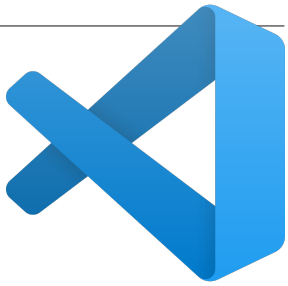


FIGURE 2.20 – Logo Vs code

Github

FIGURE 2.21 – Logo Github

C'est une plateforme open source de gestion de version et de collaboration destinée aux développeurs de logiciels. Livrée en tant que logiciel à la demande(SaaS, Software as a Service), la solution GitHub a été lancée en 2008. Elle repose sur Git, un système de gestion de code open source . Sans GitHub, l'utilisation de Git nécessite généralement un peu plus de connaissances techniques et l'utilisation de la ligne de commande. Git permet de stocker le code source d'un projet et de suivre l'historique complet de toutes les modifications apportées à ce code. Grâce aux outils qu'elle fournit pour gérer les conflits éventuels résultant des changements apportés par plusieurs développeurs, il est possible de collaborer efficacement sur un même projet.

2.6 Conclusion

Ce chapitre nous a servi à présenter la problématique traitée, et à comprendre les notions de base de la théorie des graphes, ainsi que les bases de la construction du code principal.

CHAPITRE 3

Implémentation

3.1 Introduction

Dans ce chapitre, nous allons nous concentrer sur l'implémentation de l'algorithme de détection de cliques maximales. Nous commencerons par présenter notre algorithme, suivi d'une analyse de ses performances.

3.2 Algorithme proposé pour le problème MCE

Algorithm 1 Algorithm 1

A graph G All maximal cliques of G
Calculate the degeneracy and degeneracy order of the graph G
Initialize a hash table T

for *each vertex* j *of* G **do**
  Calculate all maximal cliques of the subgraph G_j
  for *each maximal clique* K *of* G_j **do**
    Order the nodes of K based on σ_G
    Search for K in T
    if K *is already in* T **then**
      Reject K
    end
    else
      Insert the vertices of K into T
      Output K
    end
  end
end

Explication

L'algorithme principal d'énumération de cliques maximales pour un graphe non orienté qu'on a utilisé est l'algorithme de Bron Kerbosch, la complexité de cet algorithme dépend du nombre de sommets, afin d'améliorer la complexité on a utilisé une nouvelle méthode de décomposition basée sur la dégénérescence d'un graphe.Cette méthode consiste à décomposer le graphe principal en sous graphes induits G_i , $i \in [n], G[N[v_i] \cap V_i]$, tel que :

- $N[v_i]$ est l'ensemble $N(v_i) \cup v_i$ qui est défini comme le voisinage fermé de s_i .
- V_i est l'ensemble des sommets suivants, y compris lui-même selon l'ordre de dégénérescence.
- Ensuite appliquer l'algorithme Bron Kerbosch sur les sous graphes résultants, et à la fin, filtrer la liste des cliques qu'on obtient pour éliminer les cliques redondants et obtenir la liste des cliques maximales.

Cette nouvelle méthode permet de réduire le temps d'exécution de l'algorithme en diminuant la taille du graphe sur lequel on l'applique.

Algorithm 2 BronKerbosch2(R,P,X)

```
if P et X sont vides then
    R est une clique maximale
end
else
    Choisir un sommet pivot u dans P ∪ X
    for tout sommet v dans P \ N(u) do
        BronKerbosch2(R ∪ v, P ∩, X ∩ N(v))
        P ← P\ v
        X ← X∪ v
    end
end
```

L'algorithme de Bron Kerbosh avec pivot, qu'on va l'utiliser dans l'algorithme de Bron Kerbosh avec ordre de dégénérescence est le suivant :

- On initialise trois ensembles tels que :
  P : l'esemble des sommets du graphe.
  R : l'ensemble des cliques maximales du graphe.
  X : l'ensemble des sommets déjà visités.
- Si les ensembles P et X sont vides alors R est une clique maximale du graphe.
- Si ce n'est pas le cas, on choisi un sommet pivot u $\in P \cup X$ contenant les sommets restants.
- En utilisant une boucle for, on cherche tous les sommets v qui ne sont pas adjacents à u dans P.
- L'algorithme utilise un appel récursif de la même fonction en utilisant les paramètres : la réunion de l'ensemble R et le sommet v, l'intersection de l'ensemble P et les sommets adjacents à v, et l'intersection de l'ensemble X et les sommets adjacents à v.

— Par la suite, on supprime le sommet v de l'ensemble P et on l'ajoute dans l'ensemble X des sommets visités, et ainsi de suite jusqu'à l'obtention des cliques maximales du graphe.

L'algorithme ci-dessous est l'algorithme de Bron kerbosh avec ordonnancement des nœuds (c'est l'algorithme qu'on utilisera par la suite dans notre code).

Algorithm 3 BronKerbosch3(G)

Entrées : g
raphe G

$P = V(G)$
$R = \emptyset$
$X = \emptyset$
for *tout sommet v visités dans un ordre de dégénéréscence de G* **do**
  BronKerbosch2(v, $P \cap N(v)$, $X \cap N(v)$)
  P $\leftarrow P \setminus$ v
  X $\leftarrow X \cup$ v
end

On a utilisé cet algorithme parce qu'il est plus rapide que l'algorithme de Bron Kerbosh de référence (il utilise dégénérescence du graphe pour trouver les cliques maximales).

— On parcourt tous les sommets du graphe suivant son ordre de dégénérescence en utilisant une boucle for. On rappelle que l'ordre de dégénéréscence d'un graphe est un ordre sur ses sommets qui hiérarchise les sommets de degré faible. Cet algorithme peut rapidement supprimer les sommets qui ne sont pas inclus dans les cliques maximales du graphe.

— Pour chaque sommet v visité, l'algorithme appelle récursivement la fonction BronKerbosch2 avec les paramètres suivants : un ensemble contenant seulement le sommet v, l'intersection de P et l'ensemble des sommets adjacents à v, l'intersection de X et l'ensemble des voisins de v.

— On enlève le sommet v visité de l'ensemble P des sommets restants, et on l'ajoute X, et ainsi de suite.

Parcours de l'algorithme

Pour bien comprendre l'algorithme principal, on va le parcourir sur un petit graphe de 21 sommets :

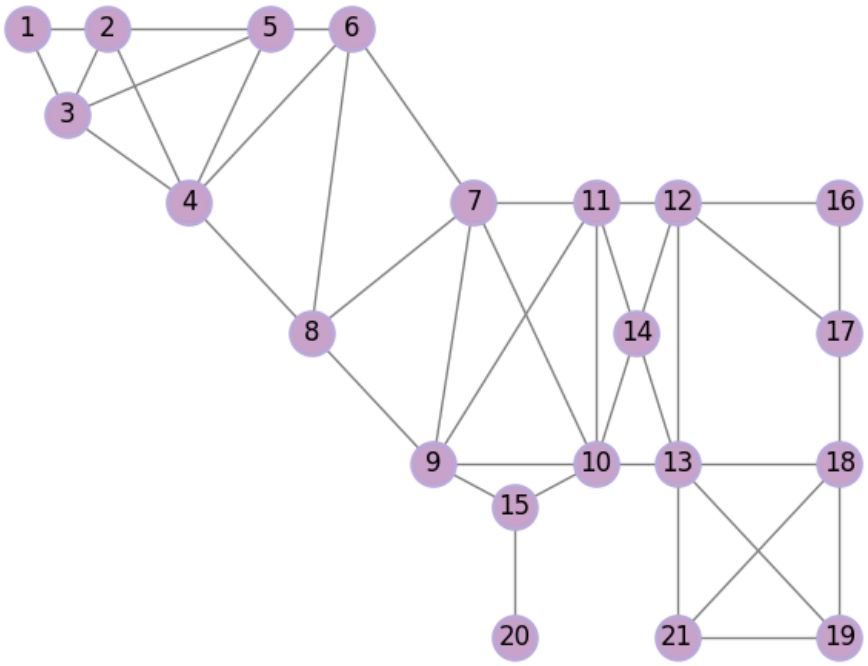


FIGURE 3.1 – Graphe non orienté de 21 sommets

Ensuite, on génère pour chaque sommet un sous-graphe G_j pour ensuite extraire toutes les cliques maximales contenues dans chaque sous-graphe. Pour ce graphe, on doit d'abord générer 21 sous-graphe G_j comme suit :

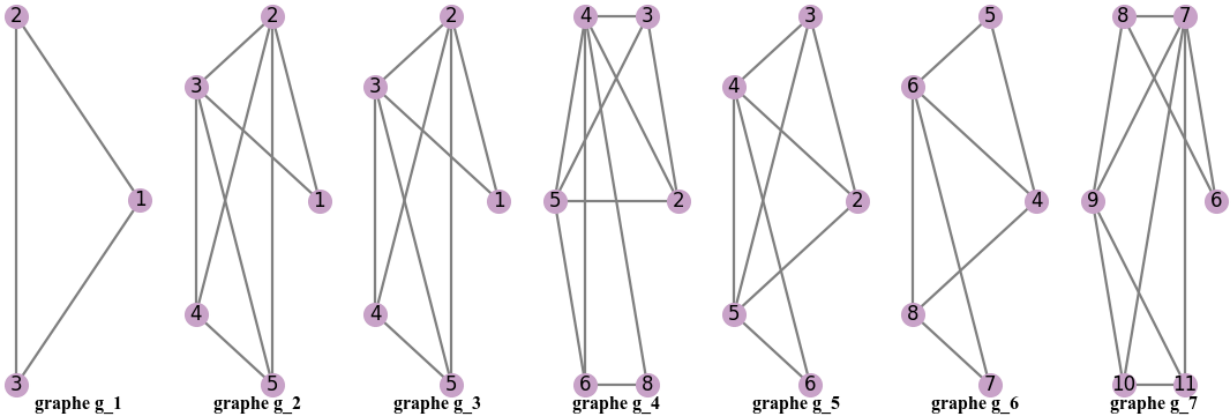


FIGURE 3.2 – Graphes g_j du graphe G

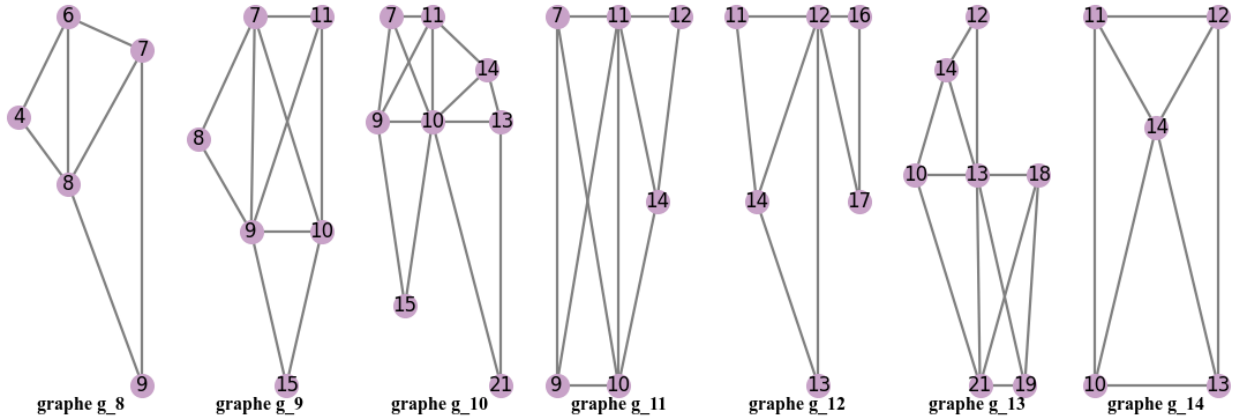


FIGURE 3.3 – Graphes g_j du graphe G (suite 1)

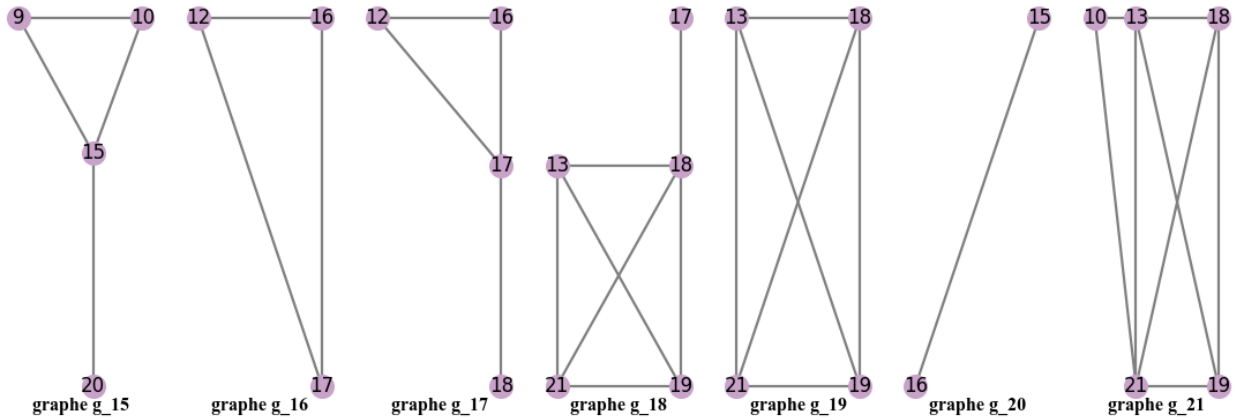


FIGURE 3.4 – Graphes g_j du graphe G (suite 2)

Après avoir extraire tous les sous-graphes G_j, $j \in 1, ..., 21$, on enlève les cliques maximales contenues dans chaque graphe G_j, on les ordonnent en fonction de leur ordre de dégénérescence et on supprime les duplications (ou si on trouve une clique maximale incluse dans une autre plus grande), on obtient à la fin de l'algorithme l'ensemble des cliques maximales suivant :

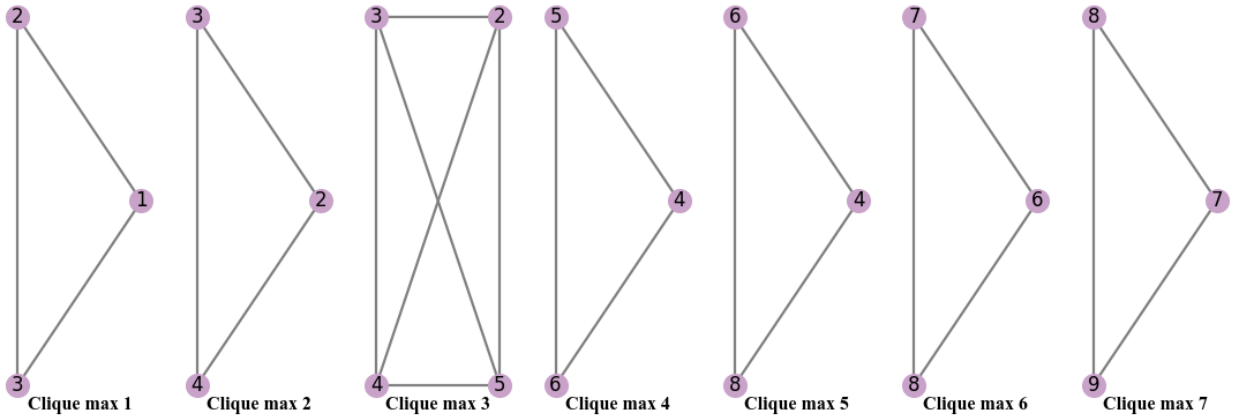


FIGURE 3.5 – Cliques maximales résultantes (partie 1)

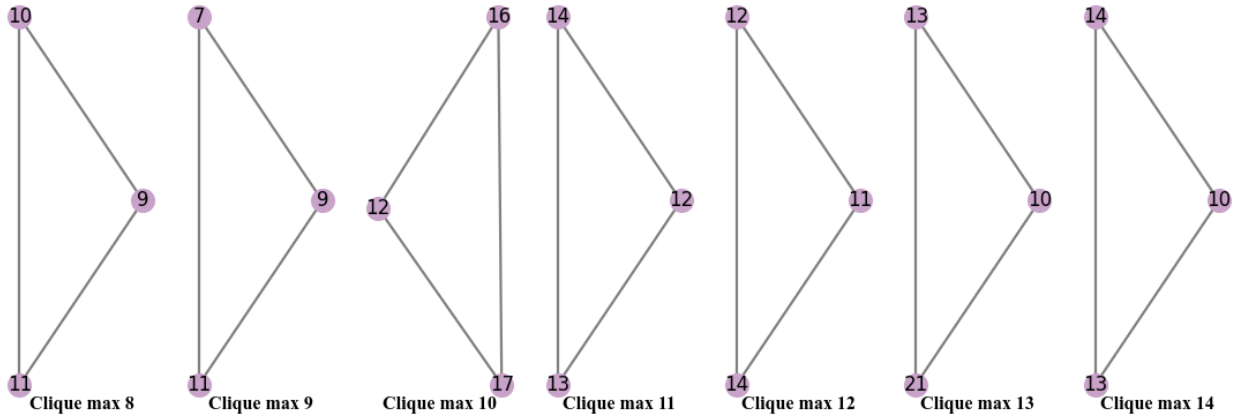


FIGURE 3.6 – Cliques maximales résultantes (partie 2)

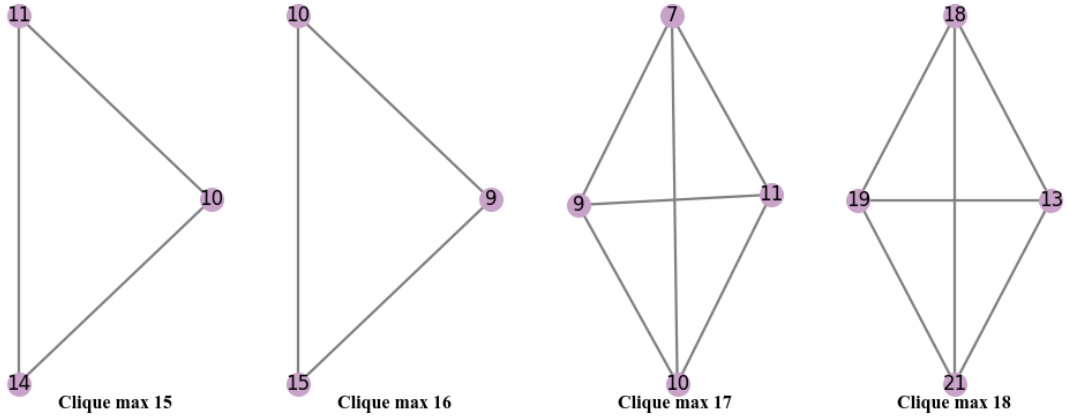


FIGURE 3.7 – Cliques maximales résultantes (partie 3)

3.3 Analyse des performances

Pour améliorer le temps de l'exécution de notre code en C++, on a ajouté des drapeaux d'optimisation lors de l'exécution du code séquentiel (version normale du code).

Parmi les flags d'optimisation qu'on a utilisé :

Alignement mémoire

Pour que les processeurs de notre machines puissent accéder rapidement et d'une manière plus performante aux données du code lors de son exécution, on utilise l'alignement en mémoire pour mieux optimiser les performances du code initial. Pour qu'on puisse aligner les données du code, on a utilisé **'-falign-loops'** pour aligner les boucles dans les programmes pour que le processeur puisse accéder d'une manière plus flexibles aux données qui vont être alignées.

La vectorisation

La vectorisation consiste à traiter plusieurs données à la fois. On va utiliser la vectorisation dans ce code pour optimiser et nettoyer notre programme car il contient quelques calculs dans les boucles, donc on peut effectuer plusieurs opérations à la fois à l'aide des instructions vectorielles. Pour vectoriser le code, on a ajouté l'option **"-ftree-vectorize"** durant l'exécution du code, pour que le processeur cherche dans le programmes les boucles qui peuvent être vectorisées, et donc il peut utiliser directement des instructions vectorielles pour exécuter le code et optimiser à la fois le temps de la compilation.

Déroulage de boucle

Le déroulage de boucle (Loop unrolling) nous permet d'exécuter toutes les itérations d'une boucle en utilisant une seule instruction qui exécute toutes les itérations à la fois. Cette technique améliore les performances de notre code. Pour utiliser `"loop unrolling"`, on a ajouté l'option **'-funroll-loops'**.En utilisant la commande **cmake**, les boucles du code seront déroulées.

Autres optimisations

On a également testé les performances du code en rajoutant les options suivantes :

— **-march=native** : elle aide à optimiser le code en utilisant les caractéristiques de la machine sur laquelle on effectue la compilation (processeurs,mémoire vive, etc).

— **-march=corei3-avx** : elle effectue les optimisations nécessaires pour le processeur Intel Core i3 en utilisant des intstructions AVX.

— Les flags d'optimisation *-Ofast*,*-O0*,*-O1*,*-O2*, et *-O3*.

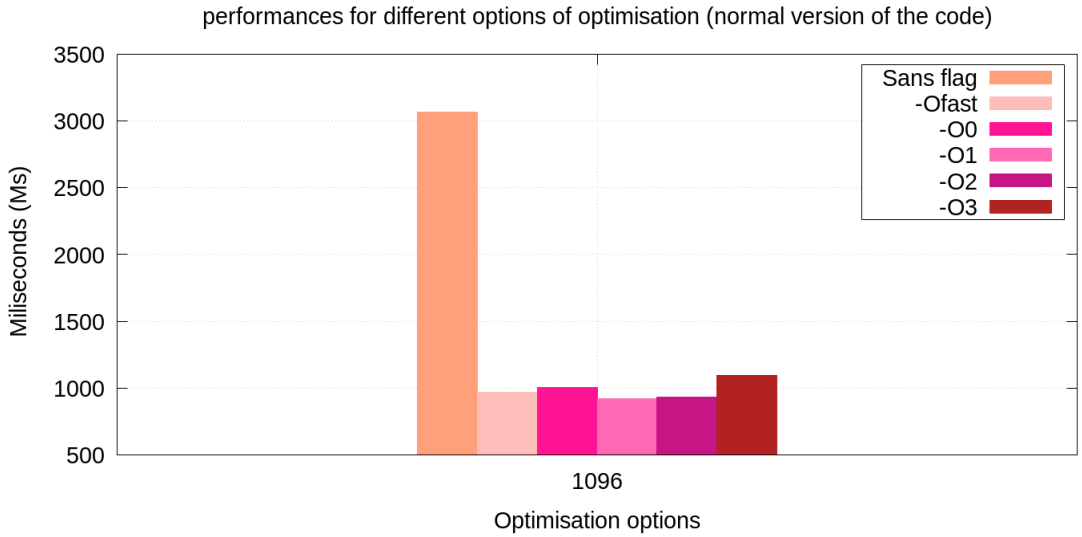


FIGURE 3.8 – Performances pour les différents Cflags d'optimisation

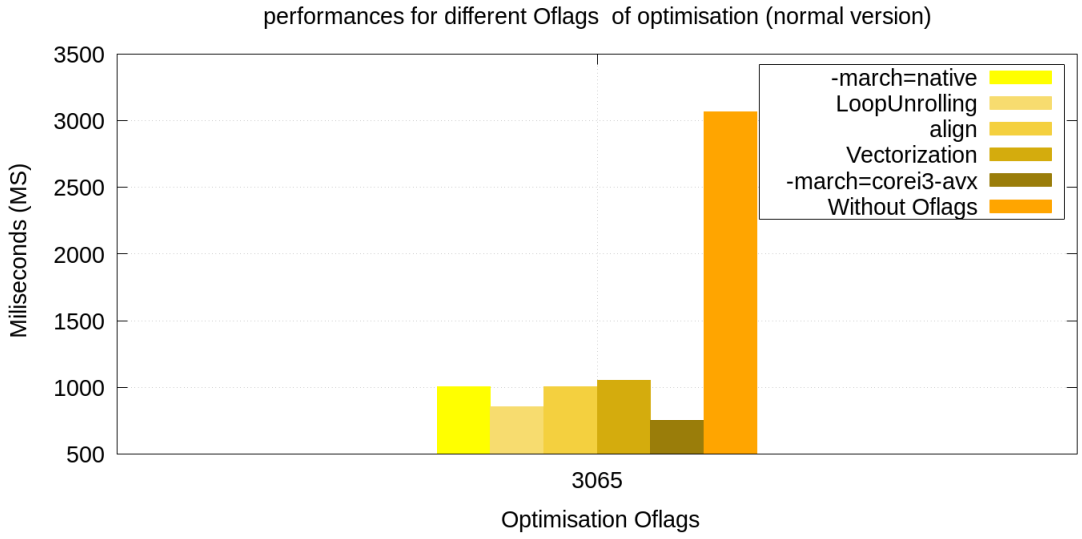


FIGURE 3.9 – Performances pour les différents Cflags d'optimisation

On remarque que les drapeaux **'-march=corei3-avx'** et **'-O1'** sont les plus performants pour la version séquentielle.

Deuxième partie

Partie parallèle

CHAPITRE 4

Parallélisation avec OpenMP

4.1 Introduction

Dans ce chapitre on va présenter notre travail sur la parallélisation et l'optimisation du code, avec une comparaison de performances avant et après l'application du parallélisme.

Parallélisme

C'est une technique informatique qui consiste à exécuter plusieurs tâches simultanément, Le but de cette technique est d'effectuer, par une machine, le plus grand nombre d'opérations dans le plus petit temps donné.

Il existe plusieurs types de parallélisme, dans notre cas nous avons utilisé le parallélisme SMP(Symmetric Multiprocessing) dans lequel plusieurs processeurs sont connectés à un bus système commun et partagent la même mémoire vive (RAM) pour exécuter des tâches en parallèle.

OpenMp

est une interface de programmation pour la programmation parallèle sur architecture à mémoire partagée SMP(avec bloc de mémoire vive qui est accédé par différentes unités de calcul), elle permet la parallélisation de tâches pour les programmes écrits en langage C, C++ ou Fortran grâce à des directives données au compilateur sous forme de '**#pragma**'.

4.2 Explication des directives utilisés

Dans notre code nous avons des fonctions qui contiennent des boucles avec un nombre d'itérations très élevé comme **find_gj()**,**calculate_degrees()** et **bron_kerbosch()**, pour cela nous avons opté pour l'utilisation des directives OpenMP qui permettent de paralléliser les boucles en C++.

#pragma omp parallel for : Utilisée pour paralléliser les boucles **for** en répartissant les itérations de la boucle sur plusieurs threads (processus léger), les itérations de la boucle sont distribuées de manière automatique et équitable entre les différents threads créés par la directive. En ajoutant la clause **shared(données à partager entre les processus)** à cette directive, on donne accès aux processus qui exécutent cette région parallèle de lire et de modifier la valeur de la variable entre parenthèses.

Explication des directives

#pragma omp critical : est une clause de synchronisation qui permet de limiter l'accès à une partie du code à un seul thread à la fois, cette partie contient généralement des modifications sur des données qui peuvent engendrer des conflits, et elle s'appelle la section critique. La directive critical garantit que les ressources partagées sont utilisées de manière cohérente et évite les conflits entre les threads.

4.3 Analyse des performances

Pour analyser les performances du code parallélisé, nous avons utilisé les mêmes options d'optimisation citées dans la partie séquentielle.

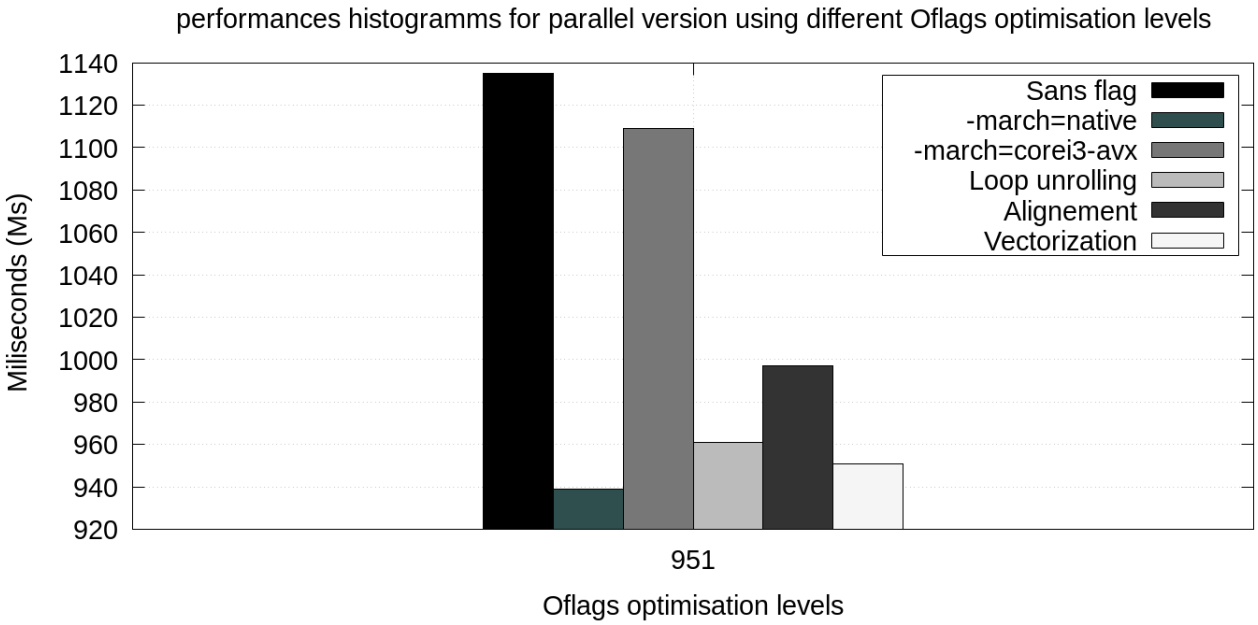


FIGURE 4.1 – Histogrammes pour différentes options d'optimisation

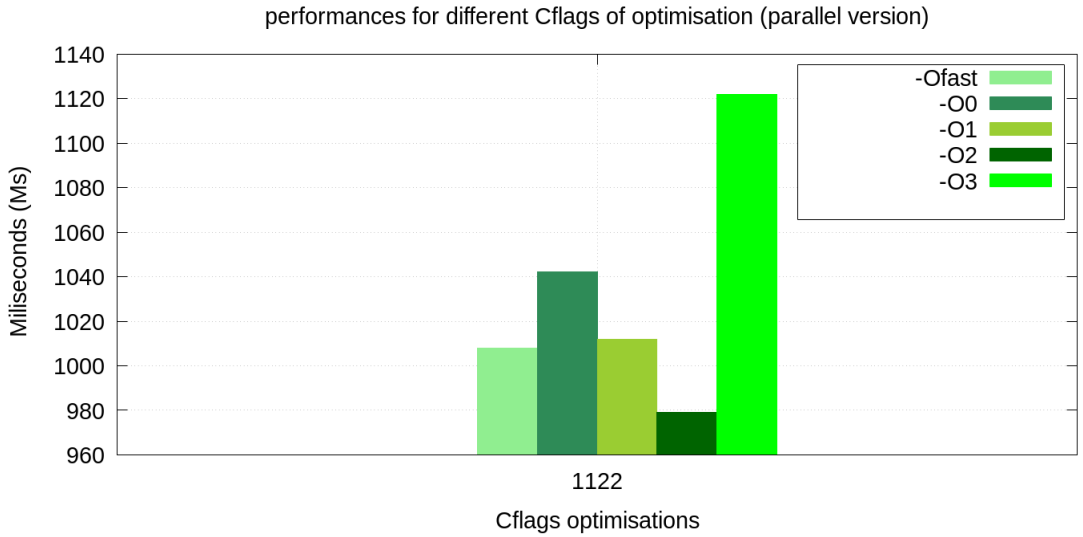


FIGURE 4.2 – Histogrammes pour différentes flags d'optimisation

On remarque que les flags d'optimisation **-march=native** et **-O2** sont les plus optimals pour la version parallèle du code.

Comparaison des résultats

Le graphe ci-dessous est une représentation des résultats obtenus en fonction de la durée d'exécution du programme entre la version initiale et la version parallélisée.

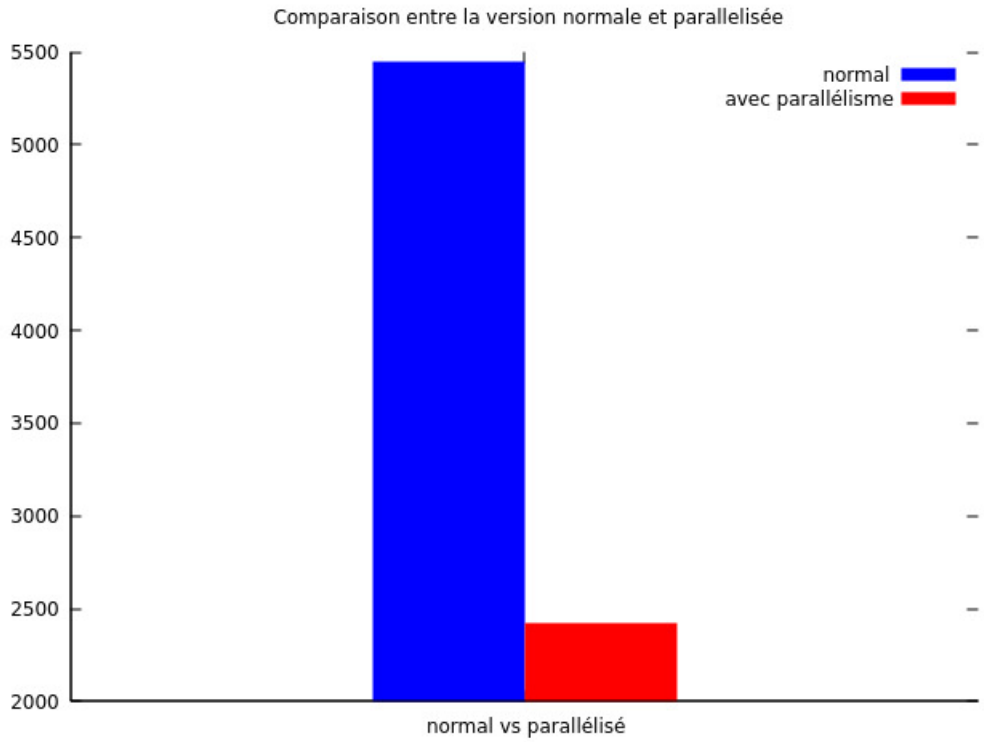


FIGURE 4.3 – Comparaison entre la version séquentielle et parallèle

L'impact du parallélisme sur cet algorithme en fonction de sa durée d'exécution est montré dans le tableau suivant :

Version initiale (ms)	Version parallélisée (ms)
5452	2420

TABLE 4.1 – Comparaison entre les deux versions de l'algorithme

Les histogrammes suivants montrent une comparaison globale entre les drapeaux d'optimisation utilisés et leurs impact sur chaque version.

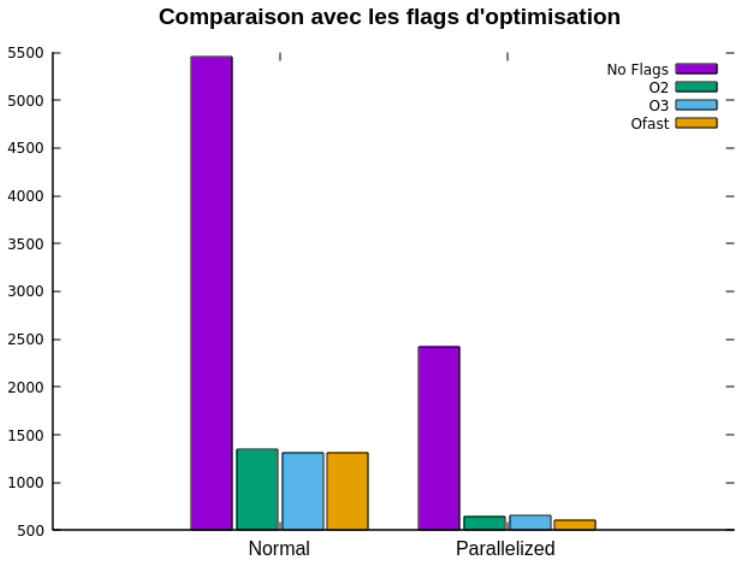


FIGURE 4.4 – Comparaison des flags entre la version normale et parallélisée

On peut bien remarquer la grande différence entre le code séquentiel et parallèle en fonction de son temps d'exécution, le drapeau **-Ofast** est bien utile en terme de performances pour le code parallèle.

CHAPITRE 5

Conclusion générale

Ce projet a comme but d'énumérer toutes les cliques maximales d'un graphe donné. Afin de pouvoir le réaliser, nous avons fait des recherches sur le problème de cliques maximales et la théorie des graphes, puis nous avons implémenté l'algorithme de Bron Kerbosch et l'algorithme principal basé sur la dégénérescence d'un graphe et enfin nous avons parallélisé le code en utilisant l'interface de programmation OpenMp.
Notre Projet pourrait être amélioré selon les perspectives suivantes :

- Ajout de mesures de performances.
- Optimisation au niveau de la mémoire.
- Utilisation d'autres algorithmes et comparer les résultats avec Bron Kerbosch.

Références

[1] URL : https://www.latex-project.org/. (Consulté : 06.04.2023).

[2] URL : https://github.com/GayathriBalakumar/Community-detection-on-social-media/blob/master/Report%20-%20Detecting%20Tight%20Communities%20and%20Friend%20Recommendation%20on%20Facebook.pdf. (Consulté : 06.12.2022).

[3] URL : https://github.com/Rem20000/AP_project/tree/main. (Consulté : 30.04.2023).

[4] Douglas B.West. *Introduction to Graph Theory - Second edition*. Pearson Education (Singapore), 2001.

[5] Hamida Seba Jocelyn Bernard. *Résolution de problèmes de cliques dans les grands graphes*. URL : https://hal.science/hal-01284640/file/resolution-de-problemes-de-cliques.pdf. (Consulté : 16.11.2022).

[6] Godfry Justo. *Hash tables*. URL : https://eng.libretexts.org/Bookshelves/Computer_Science/Programming_and_Computation_Fundamentals/Book%3A_Algorithm_Design_and_Analysis_(Justo)/04%3A_Hash_Tables_Graphs_and_Graph_Algorithms/4.01%3A_Activity_1_-_Hash_Tables. (Consulté : 28.12.2022).

[7] Jure Leskovec. *email-Eu-core network*. URL : https://snap.stanford.edu/data/email-Eu-core.html. (Consulté : 01.03.2023).

[8] George Manoussakis. "A new decomposition technique for maximal clique enumeration for sparse graphs". In : *ELSEVIER* 770.10 (24 May 2019), p. 25-33. DOI : https://doi.org/10.1016/j.tcs.2018.10.014.

[9] OpenClassrooms. *Stockez et retrouvez des données grâce aux tables de hachage*. URL : https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/19978-stockez-et-retrouvez-des-donnees-grace-aux-tables-de-hachage. (Consulté : 04.01.2023).

[10] *Oracle VM VirtualBox*. URL : Oracle%20VM%20VirtualBox%20%E2%80%94%20Wikip%C3%A9dia%20(wikipedia.org). (Consulté : 06.05.2023).

[11] *Visual Studio Code*. URL : Visual%20Studio%20Code%20%E2%80%94%20Wikip%C3%A9dia%20(wikipedia.org). (Consulté : 02.05.2023).

[12] Wikipédia. *Algorithme de Bron-Kerbosch*. URL : https://fr.wikipedia.org/wiki/Algorithme_de_Bron-Kerbosch. (Consulté : 20.11.2022).

[13] Wikipédia. *C++*. URL : https://fr.wikipedia.org/wiki/C%2B%2B. (Consulté : 29.04.2023).

[14] Wikipédia. *Clique (théorie des graphes)*. URL : https://fr.wikipedia.org/wiki/Clique_(th%C3%A9orie_des_graphes). (Consulté : 03.01.2023).

[15] Wikipédia. *Github*. URL : https://fr.wikipedia.org/wiki/GitHub. (Consulté : 04.05.2023).

[16] Wikipédia. *Linux*. URL : Linux%20%E2%80%94%20Wikip%C3%A9dia%20(wikipedia.org). (Consulté : 06.01.2023).

[17] Wikipédia. *Matplotlib*. URL : https://fr.wikipedia.org/wiki/Matplotlib. (Consulté : 03.05.2023).

[18] WIKIPÉDIA. *Numpy*. URL : https://en.wikipedia.org/wiki/NumPy. (Consulté : 29.04.2023).

[19] WIKIPÉDIA. *Python (langage)*. URL : https://fr.wikipedia.org/wiki/Python_(langage). (Consulté : 01.05.2023).