MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ MOULOUD MAMMERI, TIZI-OUZOU

FACULTÉ DES SCIENCES

DÉPARTEMENT DE MATHÉMATIQUES

MEMOIRE DE FIN DÉTUDE

Filière : Mathématiques

Spécialité : Mathématiques appliquées en sciences de gestion

Présentée par :

BOUNSIAR Rezika
AMEUR Aldjia

Sujet :

COLORATION DES GRAPHES ET SES APPLICATION DANS L'OPTIMISATION DES RESSOURCES

Devant le jury d'examen composé de :

Mr. AOUANE M.	UMMTO	Président
Mr. TALEM Dj.	UMMTO	Encadreur
Mme. AKLOUCHE F.	UMMTO	Examinatrice

Année universitaire 2024-2025

Dédicace

À la mémoire de ma chère mère
Dont l'amour, les sacrifices et les prières m'accompagnent à jamais. Ce travail est le témoignage de tout ce que tu m'as transmis.

À mon père
Pour son soutien inestimable, sa patience et ses encouragements constants, qui ont été pour moi une source de force et de persévérance.

À mes frères et sœurs et ma tante nadia
Pour leur affection, leur compréhension et leur appui indéfectible.

À mes proches et mes amis

À ma binôme AMEUR Aldjia
Avec qui j'ai partagé les efforts, les défis et les réussites de ce travail, ainsi qu'à sa famille.

Je vous dédie ce modeste travail avec tout mon respect et ma profonde gratitude.

BOUNSIAR Rezika

Dédicace

Je dédie ce mémoire à ma chère famille, à mes proches qui ont toujours cru en moi et m'ont soutenue dans les moments difficiles, ainsi qu'à ma binôme, ***Rezika BOUNSIAR****, avec qui j'ai partagé ce parcours avec complicité et courage.*

AMEUR Aldjia

Remerciements

Avant tout, nous remercions Allah le Tout-Puissant de nous avoir accordé la santé, la volonté et la persévérance nécessaires pour mener à bien ce travail.

Nos remerciements les plus sincères vont à notre encadreur, **Mr. Talem Djamel**, pour son suivi attentif, ses conseils judicieux, sa disponibilité et sa patience. Ses orientations et ses encouragements ont grandement contribué à la réalisation de ce mémoire.

Nous remercions également les membres du jury, **Mr. Aouane Mohouhand** et **Mme. AKLOUCHE Fariza**, pour avoir accepté d'évaluer ce travail.

Nous exprimons aussi notre profonde gratitude envers l'ensemble de nos enseignants du département de mathématiques, qui nous ont transmis leurs connaissances et nous ont guidés tout au long de notre parcours académique.

Nous adressons aussi nos remerciements les plus chaleureux à nos familles, pour leur soutien moral, leurs encouragements constants et leur compréhension, qui ont été une source inestimable de motivation et de force.

Enfin, nous tenons à remercier toutes les personnes qui, de près ou de loin, nous ont apporté leur aide, leurs conseils et leurs encouragements pour la réalisation de ce travail.

Résumé

Ce mémoire est consacré à l'étude de la coloration des graphes, en mettant en lumière certains de ses cas théoriques, ses aspects algorithmiques et ses applications pratiques. Dans le premier chapitre, nous présentons les notions de base de la théorie des graphes et de la complexité algorithmique, ainsi que les techniques classiques de parcours (BFS, DFS, LexBFS), qui constituent des outils essentiels d'analyse. Nous présentons ensuite l'état de l'art relatif au problème de la coloration des graphes, en définissant le nombre chromatique et en étudiant plusieurs algorithmes d'approximation tels que l'algorithme glouton, Welsh & Powell, et DSATUR. Le troisième chapitre s'intéresse à la coloration des sommets dans les graphes chordaux, où nous rappelons les propriétés structurelles de ces graphes particuliers et illustrons leur utilité à travers une application concrète à la gestion de la bande passante dans les réseaux en fibre optique. Le quatrième chapitre traite de la coloration des arêtes dans les graphes bipartis, en abordant les notions de couplage et de coloration optimale, et en présentant une application pratique dans la planification des emplois du temps.

Ainsi, ce travail met en évidence la richesse de la théorie des graphes et son importance dans la résolution et la modélisation de problèmes d'optimisation et de gestion.

Mots-clés : Théorie des graphes, Coloration, LexBFS, Graphes chordaux, Graphes bipartis, Fibre optique, Planification.

Abstract

This thesis is devoted to the study of graph coloring, highlighting its theoretical aspects, algorithmic foundations, and practical applications.

In the first chapter, we present the basic concepts of graph theory and algorithmic complexity, as well as classical traversal techniques (BFS, DFS, LexBFS), which constitute essential tools for analysis.

We then review the state of the art related to the graph coloring problem, defining the chromatic number and studying several approximation algorithms such as the Greedy, Welsh & Powell, and DSATUR algorithms.

The third chapter focuses on vertex coloring in chordal graphs, where we recall the structural properties of these particular graphs and illustrate their usefulness through a practical application to bandwidth management in fiber-optic networks.

The fourth chapter deals with edge coloring in bipartite graphs, addressing the notions of matching and optimal coloring, and presenting a practical application to timetable scheduling.

Thus, this work highlights the richness of graph theory and its importance in modeling and solving optimization and management problems.

Keywords : Graph Theory, Graph Coloring, LexBFS, Chordal Graphs, Bipartite Graphs, Fiber Optics, Timetabling.

Table des figures

Table des matières

Introduction

Depuis plusieurs décennies, la théorie des graphes s'est imposée comme un champ central des mathématiques discrètes, en raison de sa puissance de modélisation et de ses nombreuses applications dans divers domaines scientifiques et industriels. Initialement développée pour résoudre des problèmes abstraits, tels que le célèbre problème des ponts de Königsberg étudié par Euler en 1736, cette théorie s'est progressivement enrichie pour devenir un outil incontournable en informatique, en recherche opérationnelle, en économie, en gestion et même en biologie.

Un graphe constitue une structure mathématique permettant de représenter des relations ou des interactions entre entités. Les sommets (ou nœuds) représentent les entités, tandis que les arêtes traduisent les relations entre eux. Cette abstraction simple mais puissante offre un cadre naturel pour modéliser des réseaux de transport, des communications, des systèmes logistiques, des plannings ou encore des interactions sociales et biologiques.

Parmi les nombreuses problématiques étudiées dans ce domaine, la coloration des graphes occupe une place importante. Elle consiste à attribuer des «couleurs » (ou étiquettes) aux sommets ou aux arêtes d'un graphe en respectant certaines contraintes. Ce type de problème se retrouve dans des applications concrètes très variées : allocation de fréquences dans les réseaux sans fil, planification des emplois du temps, gestion de ressources partagées, optimisation des circuits électroniques, ou encore organisation de plannings en entreprise.

Un autre aspect essentiel de la théorie des graphes est l'analyse algorithmique, qui permet de déterminer l'efficacité des méthodes de résolution. Les algorithmes de parcours (BFS, DFS, LexBFS), les algorithmes gloutons et les techniques de reconnaissance de classes particulières de graphes (tels que les graphes parfaits ou les graphes d'intervalles) sont des outils puissants qui offrent des solutions efficaces à des problèmes souvent complexes, notamment dans le cadre des graphes NP-complets.

Cependant, certains problèmes de graphes, comme la coloration optimale ou le problème du voyageur de commerce (TSP), appartiennent à la classe des problèmes NP-complets, pour lesquels aucune solution polynomiale n'est encore connue.

L'étude de ces problèmes illustre le lien profond entre la théorie des graphes et la complexité algorithmique, posant la question centrale de l'informatique théorique : P = NP ?.

L'objectif de ce mémoire est d'étudier en profondeur la coloration des graphes, en mettant en lumière les aspects théoriques, algorithmiques et applicatifs. Nous nous intéresserons particulièrement :

- aux notions fondamentales de la théorie des graphes et de la complexité,
- aux différents algorithmes de parcours et de coloration,
- aux graphes triangulés et à leurs propriétés, avec une application concrète à la fibre optique,
- aux graphes bipartis et à la coloration des arêtes, avec une application à la planification des emplois du temps.

Ainsi, ce travail se veut à la fois théorique et appliqué : il met en avant la richesse des résultats mathématiques tout en illustrant leur utilité dans des situations réelles de gestion et d'optimisation.

Organisation du mémoire

Ce mémoire est structuré en quatre chapitres principaux :

- **Chapitre 1** : Introduction aux notions préliminaires nécessaires, telles que les définitions et notations de base, les graphes remarquables, les parcours dans les graphes, ainsi qu'une introduction à la complexité algorithmique.
- **Chapitre 2** : État de l'art sur le problème de la coloration des graphes, en définissant le nombre chromatique, en exposant les principaux algorithmes d'approximation (glouton, Welsh & Powell, DSATUR), et en discutant des cas particuliers comme les graphes planaires et triangulés.
- **Chapitre 3** : Étude de la coloration des sommets dans les graphes triangulés, en présentant leurs propriétés structurelles et en illustrant par une application concrète au problème de la fibre optique.
- **Chapitre 4** : Coloration des arêtes dans les graphes bipartis. Après avoir rappelé leurs définitions et propriétés, nous présentons les résultats théoriques sur la coloration optimale et proposons une application au problème de l'emploi du temps.

Pour résumé, ce mémoire illustre comment une théorie mathématique abstraite peut fournir des solutions élégantes et efficaces à des problèmes concrets de gestion et d'optimisation, démontrant ainsi l'importance de la théorie des graphes dans les sciences appliquées contemporaines.

Chapitre 1

Préliminaires

Ce chapitre introductif a pour objectif de poser les bases théoriques et conceptuelles nécessaires à l'étude approfondie des problèmes de coloration de graphes. Nous commencerons par définir les notions fondamentales de la théorie des graphes, accompagnées des exemples illustratifs pour en faciliter la compréhension. Ensuite, nous aborderons des concepts clés en algorithmique et optimisation, notamment les problèmes d'optimisation combinatoire, la notion d'algorithme, et les classes de complexité (P, NP, NP-complétude), avec un accent particulier sur leur lien avec le problème de la coloration. Enfin, nous présenterons les principaux parcours de graphes (BFS, DFS et LexBFS), qui joueront un rôle essentiel dans les algorithmes étudiés ultérieurement.

Ce préambule permettra d'unifier le langage et les outils utilisés dans les chapitres suivants, tout en offrant une vision claire des enjeux théoriques et pratiques associés à la coloration des graphes.

1.1 Notion de graphe

Un **graphe non orienté** G ou simplement un graphe (graph) est défini par un couple de deux ensembles $G = (V_G, E_G)$,où $V_G = \{v_1, v_2, , \ldots, v_n\}$ est un ensemble de **sommets** (vertices) de G qui ne doit pas être vide, et $E_G = \{e_1, e_2, \ldots, e_m\}$ est une relation binaire symétrique sur V_G et dont les éléments sont appelées **arêtes** (edges). S'il n'y a pas le risque de confusion, on écrit simplement $G = (V, E)$.

Une arête e est définie par un couple non ordonné de sommets $\{u, v\}$ qui sont ses extrémités (endpoints)) ; elle est notée $e = \{u, v\}$ ou simplement $e = uv$. Puisque la relation binaire est symétrique, les notaions uv et vu désignent la même arête.

Exemple 1.1. Considérons un groupe de six personnes, Adam, Bachir, Celina,

Dalila, farid et Hamid. Le tableau ci-dessous représente la relation d'amitié entre ces individus :

La personne	ses ami(e)s
Adam	Bachir, Celina
Bachir	Adam, Celina, Dalila, Hamid
Celina	Adam, Bachir, Hamid
Dalila	Bachir, Farid, Hamid
Farid	Dalila, Hamid
Hamid	Bachir, Celina, Dalila, Farid

Il est claire que cette relation d'amitié est une relation binaire sur l'ensemble des six personne. Ainsi, cette relation peut être représentée par un graphe de la manière suivante :

— chaque personne est représentée par un point (ce point est désigné par la première lettre du prénom de cette personne), donc $V = \{a, b, c, d, f, h\}$;

— deux personnes qui sont amies, seront liées par une arête, donc $E = \{ab, ac, dh, bd, df, bc, bh, ch, fh\}$.

Le graphe $G = (V, E)$ associé à la relation binaire ci-dessus est donné par la figure 1.1).

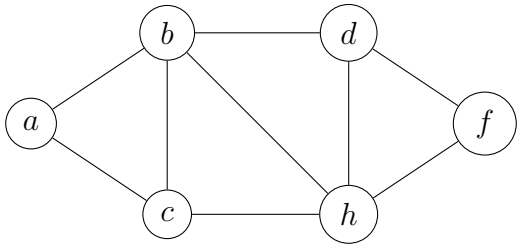


FIGURE 1.1 – Graphe de la relation d'amitié

Deux sommets distincts de G sont **adjacents** (adjacent vertices) s'ils sont reliés par une arête. De même, deux arêtes distinctes sont considérées comme adjacentes si elles partagent une même extrémité. Dans la figure ci-dessus, deux sommets sont adjacents si et seulement si il sont amis.

Une arête de la forme $e = uu$ (où un sommet est relie à lui-même) est appelée **boucle** (loop).

Un graphe est un **multigraphe** (multigraph) s'il autorise plusieurs arêtes distinctes entre une même paire de sommets.

Un graphe est dit **simple** (simple graph) s'il vérifie simultanément deux conditions : il est sans boucle et sans multi-arêtes. Autrement dit, il est sans boucle et entre deux sommets il y a au plus une arête.

Remarque 1.1. Dans la suite de ce manuscrit, tout les graphe seront simples et fini, c'est-à-dire ayant un nombre fini de sommets.

Deux sommets adjacents sont dits **voisins**. Le voisinage ouvert (open neighbors) d'un sommet x est l'ensemble des sommets $N(x) = \{y \quad /xy \in E\}$. L'ensemble des sommets $N[x] = N(x) \cup \{x\}$ est le voisinage fermé (closed neighbors) de x. Le **degré d'un sommet** x dans le graphe G est un entier noté $d_G(x)$, qui est égale au nombre d'arêtes incidentes à x. Notons que le degré d'un sommet x est égale au nombre de ses voisins, c'est-à-dire, $d(x) = |N(x)|$.

Le nombre de sommets dans un graphe G, désigné par n, est appelé l'ordre de G, et le nombre d'arêtes, désigné par m, est appelé la taille de G. L'ordre et la taille d'un graphe simple sont reliés par la formule des degrés suivante :

$$\sum_{v \in V} d(v) = 2|E| = 2m$$

Le degré maximum d'un graphe $G = (V, E)$ est noté Δ et il est calculé comme suit :

$$\Delta(G) = \max\{d(x), x \in V\}$$

D'une manière symétrique on définit le degré minimum, noté $\delta(G)$, par la formule suivante :

$$\delta(G) = \min\{d(x), x \in V\}$$

Dans le graphe de la figure 1.1, on a $\Delta = 4 = d(h)$; $\delta(G) = 2 = d(a)$.

1.2 Représentation informatique d'un graphe non orienté

L'idée de représenter des relations entre objets remonte au mathématicien **Leonhard Euler**, qui a posé les bases de la théorie des graphes en 1736 avec le problème des ponts de Königsberg. Cependant, la représentation par une machine des graphes s'est développée plus tard, principalement au **XX^{e} siècle**, avec l'essor de l'**algèbre linéaire** et des **algorithmes sur les graphes**.

Les deux structures les plus utilisées sont la *matrice d'adjacence* et la *liste d'adjacence*. Chacune de ces représentations présente des avantages et des inconvénients en termes de mémoire et de temps d'exécution des algorithmes. Ces deux outils sont très utiles pour analyser, modéliser et résoudre des problèmes sur les graphes, que ce soit en informatique, en réseau, en transport ou dans les sciences sociales.

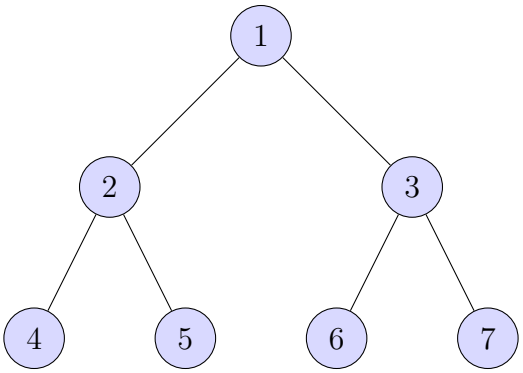
1.2.1 Représentation par une matrice d'adjacence

Soit $G = (V, E)$ un graphe non orienté avec $|V| = n$. La matrice d'adjacence de G est une matrice carrée $A = (a_{ij})_{1 \leq i,j \leq n}$ telle que :

$$a_{ij} = \begin{cases} 1 & \text{si } (i,j) \in E, \\ 0 & \text{sinon.} \end{cases}$$

Exemple 1.2. Considérons le graphe G à 7 sommets $V = \{1, 2, 3, 4, 5, 6, 7\}$ et d'arêtes :

$$E = \{(1,2), (1,3), (2,4), (2,5), (3,6), (3,7)\}.$$



La matrice d'adjacence associée au graphe G est :

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

1.2.2 Représentation par une liste d'adjacence

La liste d'adjacence consiste à associer à chaque sommet $v \in V$ la liste de ses voisins. C'est une représentation plus compacte, notamment pour les graphes creux (sparse graphs).

Exemple. Pour le graphe G :

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

— $4 : \{2\}$
— $5 : \{2\}$
— $6 : \{3\}$
— $7 : \{3\}$

1.3 Sous-graphe

Soit $G = (V(G), E(G))$ un graphe, où $V(G)$ est l'ensemble des sommets et $E(G)$ est l'ensemble des arêtes.

On appelle **sous-graphe** de G tout graphe $H = (V(H), E(H))$ vérifiant les deux conditions suivantes :

$$V(H) \subseteq V(G) \quad ; \quad E(H) \subseteq E(G)$$

On distingue plusieurs types de sous-graphes : sous-graphe obtenu par suppression d'arêtes, sous-graphe obtenu par suppression des sommets et sous-graphe obtenu par suppression des arêtes et des sommets.

Graphe partiel

On dit que H est un **Graphe partiel** (spaning graph) d'un graphe G si

$$V_H = V_G \quad ; \; E_H \subseteq E_G$$

On dit aussi, dans ce cas, H est un graphe **partiel** de G.

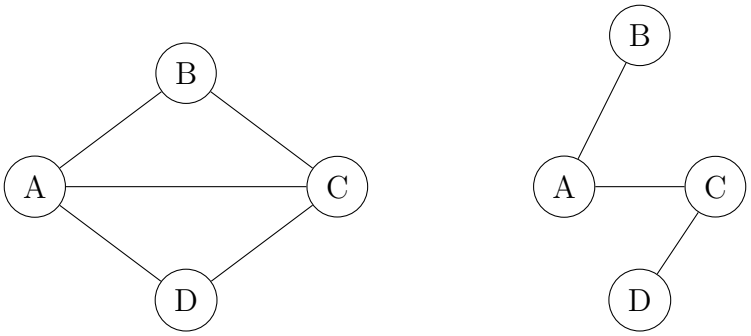


FIGURE 1.2 – Graphe G et son graphe partiel H

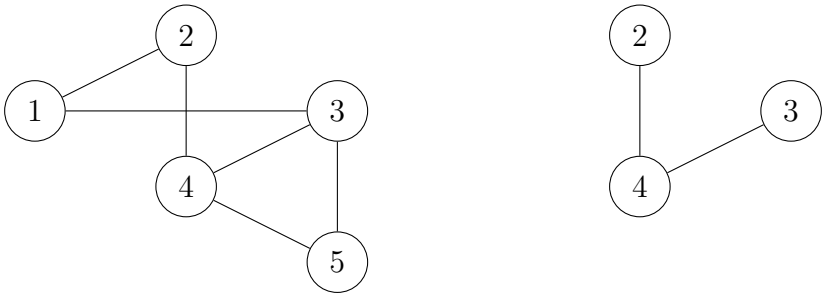


FIGURE 1.3 – Un graphe G et un sous-graphe induit par 2,3 et 4

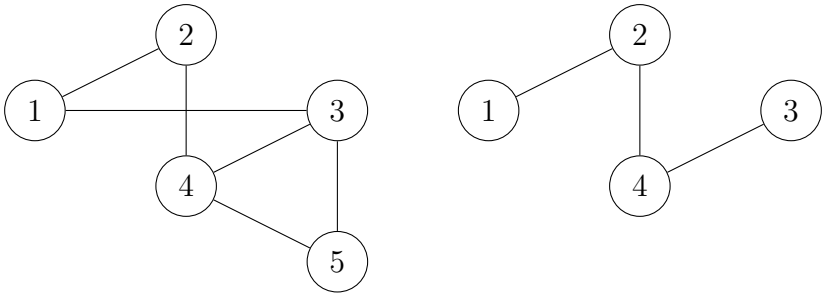


FIGURE 1.4 – Un graphe G et un sous-graphe partiel défini par les sommet 1, 2,3 et 4 et les arêtes 12, 24 et 34.

Sous-graphe induit

On dit que H est un **Sous-graphe induit** (induced subgraph) d'un graphe G s'il existe un ensemble de sommets tel que $V_H = S$ et E_H est l'ensemble des arêtes de G ayant leurs extrémités dans S. On écrit $H = G_S = (S, E_H)$.

Sous-graphe partiel

On dit que H est un **Sous-graphe partiel** si H n'est ni partiel ni induit, donc H est obtenu par la suppression des sommets et des arêtes, et on a dans ce cas :

$$V_H \subset V_G \quad ; \; E_H \subset E_G$$

Ainsi, le sous-graphe partiel H est obtenu par suppression des sommets et des arêtes.

1.4 Connexité

Une **chaîne** (path) entre un sommet x et un sommet y est définie comme une séquence ordonnée de sommets : $\alpha = (x_1, x_2, \ldots, x_k)$ telle que pour chaque i,

les sommets x_i et x_{i+1} sont **adjacents**, c'est-à-dire, il existe une arête entre eux. Autrement dit

$\alpha = (x_1, x_2, \ldots, x_k)$ est une chaine si et seulement si

$$\forall i = 1, ..., k-1, x_i x_{i+1} \in E$$

Si dans α les sommets x_k et x_1 sont adjacents, alors α est dit un **cycle**.

Une chaîne (resp. cycle) est dite **élémentaire** si elle ne passe pas deux fois par le même sommet. En d'autres termes, tous les sommets traversés sont distincts. Une chaîne (resp. cycle) élémentaire qui passe par tous les sommets du graphe G est appelée chaîne (resp. cycle) **hamiltonien**.

Un graphe G est **connexe** (connected graph) si entre deux sommets quelconques, il existe une chaine allant de l'un vers l'autre. Un graphe **non connexe** est constitué par plusieurs **composantes connexes**. Une composante connexe dans un graphe non connexe est un sous graphe induit connexe et maximal, c'est à dire n'est pas inclus dans un autre sous graphe induit connexe.

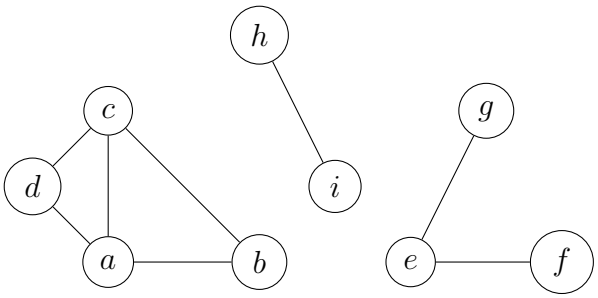


FIGURE 1.5 – Graphe non connexe avec trois composantes connexes :
$C_1 = \{a, b, c, d\}, C_2 = \{e, f, g\}$ et $C_3 = \{i, h\}$.

1.5 Graphes remarquables

Graphe régulier

on dit qu'un graphe $G = (V, E)$ est k-régulier si pour tout sommet x, on a $d(x) = k$. Les graphes k-réguliers sont caractérisé par la propriété suivante :

Proposition 1.1. *Si G est un graphe k-régulier ayant n sommets et m arêtes, alors on : $m = \frac{kn}{2}$. Un exemple de graphe 3-régulier est donné par la figure 1.6*

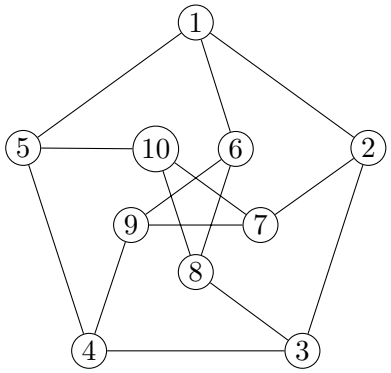


FIGURE 1.6 – Graphe 3-régulier

Graphe complet

Un graphe $G = (V, E)$ est dit complet si chaque sommet du graphe est adjacent à tous les autres sommets, c'est-à-dire

$$\forall x \in V,\ \forall y \in V,\ x \neq y \Rightarrow \{x, y\} \in E$$

Un graphe complet ayant n sommets est noté K_n. Il contient exactement :

$$|E| = \frac{n(n-1)}{2} \quad \textit{arêtes}$$

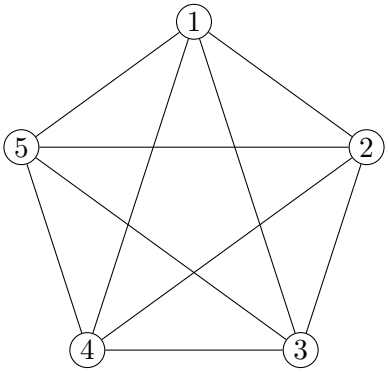
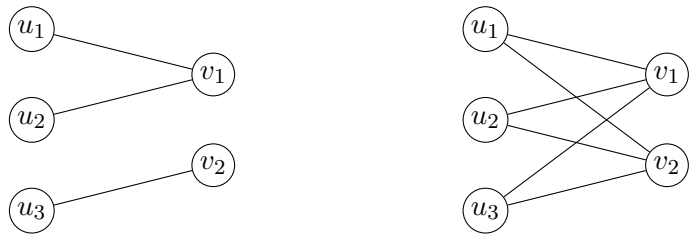


FIGURE 1.7 – Graphe complet K_5

Graphe biparti

Un graphe $G = (V, E)$ est ***biparti*** *si ses sommets peuvent être partitionnés en deux ensembles disjoints X et Y , de sorte que toute arête e possède une extrémité*

dans X et l'autre extrémité dans Y (Dans un graphe biparti, deux sommets de la même classe ne sont pas adjacents). On notera $G = (X, Y, E)$. Un graphe biparti $G = (X, Y, E)$ avec $|X| = p$, $|Y| = q$ et dans lequel $\forall x \in X, \forall y \in Y$, $xy \in E$ est appelé graphe ***biparti complet****, on le note par $K_{p,q}$. La figure 1.8 donc un exemple d'un graphe biparti et d'un graphe biparti complet.*



Biparti non complet $K_{3,2}$

FIGURE 1.8 – Graphe biparti à gauche et graphe biparti complet à droite

Graphe d'intervalle

Un ***graphe d'intervalles*** *est un graphe dont les sommets peuvent être associés à un ensemble d'intervalles sur une droite réelle, de telle sorte que deux sommets sont adjacents si et seulement si leurs intervalles correspondants se chevauchent. Autrement dit, un graphe $G = (V, E)$ est un graphe d'intervalles s'il existe une famille d'intervalles $\{I_v = [a_v, b_v] \mid v \in V\}$ telle que :*

$$(u, v) \in E \quad \Leftrightarrow \quad I_u \cap I_v \neq \varnothing.$$

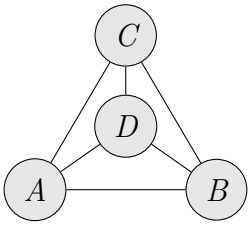
Les graphes d'intervalles constituent une sous-classe importante des graphes ***chordaux*** *et interviennent dans de nombreux problèmes de planification, d'ordonnancement et de gestion de ressources.*

Graphe planaire

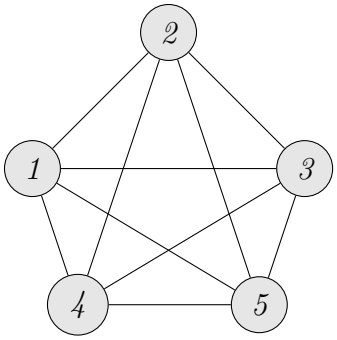
Un graphe $G = (V, E)$ est dit planaire s'il peut être dessiné dans le plan de telle sorte que ses arêtes ne se croisent pas en dehors de leurs extrémités. Autrement dit, il existe une représentation de G dans le plan où deux arêtes ne se coupent jamais, sauf si elles partagent un sommet.

Par exemple, le graphe complet à 4 sommets, K_4, est un graphe planaire, car on peut le représenter sans croisement d'arêtes. En revanche, K_5, le graphe complet

à 5 sommets, n'est pas planaire. Nous avons aussi le graphe complet biparti $K_{3,3}$ qui n'est pas planaire.



Graphe K_4 (planaire)



Graphe K_5 (non planaire)

Une caractérisation fondamentale pour les graphes planaires est donnée par le théorème de ***Kuratovsky*** *suivant :*

Théorème 1.2. *[3] Un graphe G est planaire si et seulement si il ne contient pas un sous-graphe homéomorphe à K_5 ou $K_{3,3}$.*

1.6 Parcours dans les graphes

1.6.1 Parcours de graphes

Les parcours de graphes sont des algorithmes qui, à partir d'un sommet source (ou origine) s arbitraire, permettent de visiter et marquer tous les sommets un par un. Au cours du parcours, un sommet est marqué seulement s'il est voisin d'un sommet déjà marqué. Á la fin de la procédure, les sommets accessibles seront ordonnés ou numérotés selon leur ordre de visite. Autrement dit, l'ordre de visite est un ordre total sur les sommets accessibles. Bien sûr, les algorithmes de parcours sont utilisés dans les graphes orientés et non orientés et aussi dans les posets. En théorie des graphes par exemple, on utilise souvent les parcours pour la reconnaissance des classes de graphes : la reconnaissance des graphes bipartis, des graphes sans circuit... . Ils sont aussi utilisés pour calculer une solution optimale

dans de nombreux problèmes d'optimisation combinatoire, problème de l'arbre de poids minimum (l'algorithme de ***Prim****), problème de cheminement (l'algorithme de Dijkstra), problème de flot (l'algorithme de* ***Belleman*** *et* ***Ford****) ... . Parfois, un simple algorithme glouton appliqué aux sommets énumérés par un algorithme de parcours peut être derrière la reconnaissance d'une classe de graphe particulière et la résolution de nombreux problèmes sur cette même classe comme c'est le cas avec les graphes triangulés, d'intervalles et de comparabilité ....[4, 17].*

En principe, il existe deux types de parcours fondamentaux, le parcours en largeur (BFS, pour Breadth First Search) et le parcours en profondeur (DFS, pour Depth First Search). Soit un graphe $G = (V, E)$ d'ordre n et un sommet origine s.
Parcours en profondeur
Le parcours DFS utilise la stratégie suivante : en arrivant à un sommet u il traite l'un de ses voisins qui n'est pas encore visité, soit le sommet v. Avant de traiter le voisin suivant de u, il traite d'abord tous les descendants du sommet v. Le même traitement sera appliqué à tous les sommets accessibles depuis la source. Pour cela, on utilise la structure de données ***pile*** *(stack) dont le principe est "dernier arrivé, premier sorti" (LIFO, pour Last In First Out).*

Parcours en largeur
Le parcours BFS visite et énumère systématiquement tous les sommets accessibles depuis s et calcule les distances entre s et ces sommets comme suit :
pour $k \geq 1$, le parcours BFS visite et énumère tous les sommets situés à une distance k de s avant de passer aux sommets dont la distance est $k + 1$. Dans ce parcours, on utilise la structure de données ***file*** *(queue) dont le principe est "premier arrivé, premier sorti" (FIFO, pour First In First Out).*

Remarque 1.2.
1. S'il existe toujours des sommets inaccessibles depuis le sommet origine s, on prend un sommet non visité comme une nouvelle origine, à partir duquel on démarre à nouveau le parcours, jusqu'à ce que tous les sommets soient visités.
2. Dans les parcours BFS et DFS, les arêtes empruntées pour découvrir de nouveaux sommets non marqués, voisins des sommets déjà marqués, constituent une arborescence de racine s contenant tous les sommets accessibles depuis s. Plus précisément, elle constitue une arborescence des plus courtes distances de s aux sommets accessibles.

Pour plus de détails sur ces deux parcours de graphes, nous proposons de consulter la référence [14].

Dans la suite de ce mémoire, nous nous intéressons à un type de parcours plus particulier, le ***parcours en largeur lexicographique*** *(ou LexBFS, pour Lexicographic Breadth First Search). LexBFS a été introduit en 1976 par ROSE et al [17] pour la reconnaissance d'une classe de graphes particulière, les graphes triangulés. Au cours des premières années de son introduction, LexBFS n'a pas été trop popularisé, mais à partir des années 1990, ses applications n'ont pas cessé de se multiplier (voir [26, 27, 28]).*

Définition 1.1. Soient $L_1 = (a_1, a_2, \ldots, a_k)$, $L_2 = (b_1, b_2, \ldots, b_t)$ deux vecteurs dont les coordonnées sont des entiers naturels. On dit que L_1 est inférieur lexicographiquement à L_2 et on écrit $L_1 <_{lex} L_2$ si et seulement si

1. Il existe $j \leq \min(k, t)$ tel que $a_i = b_i$ pour $i < j$ et $a_j < b_j$, ou
2. $k < t$ et $a_i = b_i$ pour $i = 1, \ldots, k$

Le vecteur obtenu par la concaténation d'un entier a à la fin d'un vecteur $L = (a_1, a_2, \ldots, a_k)$ est $L + a = (a_1, a_2, \ldots, a_k, a)$.

Dans le parcours LexBFS, on énumère les sommets de G de n à 1 à partir d'un sommet origine s comme suit : on associe à chaque sommet v une étiquette $L(v)$ qui est un vecteur. Au départ, $L(s) = n$ et pour $v \neq s$, $L(v) = \emptyset$. Pour i allant de n à 1, on assigne l'entier i au sommet v ayant une étiquette maximale lexicographiquement, et pour tout sommet w, voisin de v, non encore numéroté, on concatène l'entier i à la fin de l'étiquette $L(w)$. Sur la figure 1.9, nous montrons comment dérouler le parcours LexBFS sur un graphe.

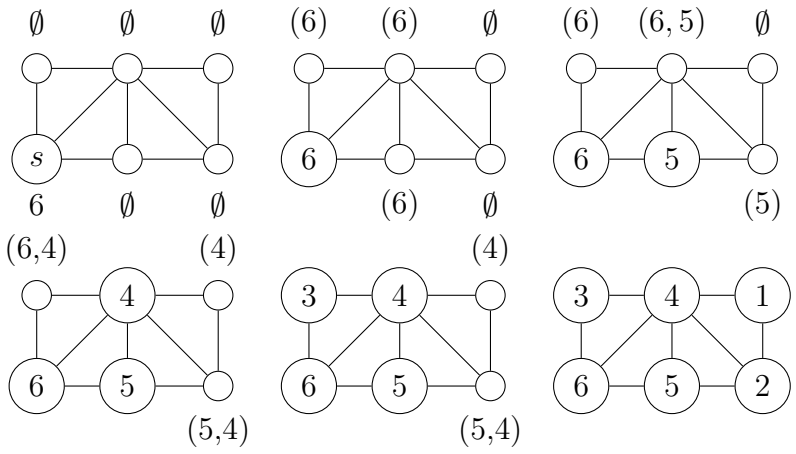


FIGURE 1.9 – Graphe ordonné par LexBFS.

1.7 Introduction à la complexité algorithmiques

La théorie de la complexité est une branche commune des mathématiques et de l'informatique qui s'intéresse à l'étude de la difficulté intrinsèque (naturelle) des **problèmes algorithmiques**, problèmes dont la solution est calculée par un algorithme ; et qui vise à classer ces problèmes en fonction de cette difficulté. Il y a deux types de complexités

- la **complexité temporelle**, donnée par le temps que met un algorithme à résoudre le problème ;
- la **complexité spatiale**, donnée par l'espace mémoire qu'utilise un algorithme au cours de son exécution.

Il faut noter que ces deux complexités sont exprimées indépendamment de la puissance des ordinateurs et des langages de programmation utilisés.

1.7.1 Problèmes de décisions et Problèmes d'optimisations

Définition 1.2. Un **problème** est une question posée sur un ensemble d'éléments appelés les **entrées** (ou les **instances**) de ce problème. Un problème est défini par trois parties : son nom, l'une de ses instances et la question à laquelle on doit répondre.

Exemple 1.3. Soient les deux problèmes suivants :

1. — Nom : Problème de primalité
   — Instance : un entier N ;
   — Question : N est-il premier ?
2. — Nom : Problème de la clique
   — Instance : Un graphe $G = (V, E)$;
   — Question : Trouver une clique de G avec un maximum possible de sommets (ou une clique maximum).

Ainsi, les instances du problème de primalité sont les entiers naturels et les instances du second problème sont les graphes. Pour répondre à la question posée, il faut fixer une instance : pour $N = 9$ la réponse est "Non", pour $N = 11$, la réponse est "Oui". Pour le second problème, et pour un graphe G qui est un arbre avec au moins une arête (par exemple), la clique maximum contient deux sommets....
Nous donnons ci-dessus de types de problèmes auxquels nous nous intéressons dans toute la suite de ce chapitre.

Définition 1.3. On appelle problème d'**optimisation combinatoire** tout problème consistant à chercher un minimum (resp. un maximum) x^* d'une certaine application f définie sur un ensemble fini X. La fonction f est appelée **fonction objectif**, et l'ensemble X est appelé **ensemble des solutions réalisables**.

Le problème de la clique définie plus haut est un problème d'optimisation combinatoire. En effet, l'ensemble des solutions réalisables est l'ensemble de toutes les cliques maximales $\mathcal{C}$ contenues dans l'instance G, la fonction objectif est celle qui associe à toute clique son cardinal, c'est-à-dire la fonction f définie par :

$$\begin{aligned} f :&\mathcal{C} \longrightarrow \mathbb{N} \\ &C \longmapsto f(C) = |C| \end{aligned}$$

Ainsi, on cherche une clique C^* vérifiant

$$f(C^*) = \max\{|C|, C \in \mathcal{C}\}$$

Définition 1.4. Un problème de **décision** est un problème pour lequel on répond par un oui ou par un non. Par exemple, l'entier 21 est-il premier ? La réponse est non ! Ainsi le problème de primalité ci-dessus est un exemple de problème de décision.

Remarque 1.3. Notons que

1. un problème de décision peut être vu comme une application définie sur un ensemble X et à valeur dans $\{0, 1\}$ telle que : $f(x) = 1$ si et seulement si la réponse à la question posée est "oui". Le problème de primalité peut être représenté par l'application

   $$\begin{aligned} f :&\mathbb{N} \longrightarrow \{0, 1\} \\ &N \longmapsto f(N) \end{aligned}$$

   avec $f(N) = 1$ si et seulement si N est premier.
2. pour un problème d'optimisation, on peut lui associer un problème de décision. Par exemple, le problème de décision associé au problème de la clique est :
   — *PROBLÈME DE LA CLIQUE*
   — **Instance** Un graphe G et un entier naturel k.
   — **Question** G posséde-t-il une clique de cardinalité k ?.

1.7.2 Complexité d'un algorithme

La résolution d'un certain problème de décision ou d'optimisation se fait à l'aide d'un algorithme.

Un **algorithme** est une sequence finie d'opérations de calcul élémentaires qui prend en entrée toute donnée I d'un problème donné P et produit en sortie un résultat *A(I)*.

Á tout algorithme correspond deux types de complexité, sa complexité temporelle et sa complexité spatiale. Mais dans ce chapitre, nous nous limitons à la complexité temporelle

Définition 1.5. La **complexité temporelle** d'un algorithme correspond au nombre d'instructions élémentaires : opérations arithmétiques ($+$, $\times$ ...), comparaisons ($\leq, \geq$), affectations ($=$), de testes d'adjacences ..., effectuées au cours de son exécution.

La complexité d'un algorithme est le nombre total des opérations élémentaires qu'effectue cet algorithme pendant toute son exécution. Ce nombre est une fonction en la taille des instances du problème considéré. Par exemple, chercher une clique de taille maximale dans un graphe avec $n = 10$ sommets n'a pas la même difficulté que de la chercher dans un graphe avec $n = 100$ sommets.

Définition 1.6. Si I est une instance d'un problème P, on appelle la taille de I, le paramètre qu'on note $taill(x)$ ou $|x|$ et qui désigne généralement le nombre (ou la quantité) de cases mémoires nécessaires pour sa représentation.

Exemple 1.4. La taille de l'instance $N = 19$ du problème de primalité est 5. En effet, dans la base binaire $(19)_2 = 2^4 + 0 \times 2^3 + 0 \times 2^2 + 2 + 1 \times 2^0 = 10011$, et donc il en faut 5 cases mémoire pour la représentation de 19 par un ordinateur. Quand une instance est un graphe, dans ce cas ça taille est donnée par les paramètres n ou m qui sont respectivement le nombre de sommets et d'arêtes de ce dernier.

Notons aussi que le temps d'exécution (ou la complexité temporelle) d'un algorithme peut varier sur des instances de même taille. En effet, pour chercher un élément dans un tableau de n nombres, on peut le faire par un algorithme qui va exécuter les opérations de comparaisons. Il est claire que l'élément cherché peut être trouvé à la premier position, donc la complexité dans ce cas est 1 (on dit la complexité est constante) ; comme on peut le trouver dans la dernière position, et dans ce cas il faut exécuter n tests de comparaisons, c'est à dire la complexité temporelle est n. Ainsi pour que notre calcul de la complexité soit fiable, on définit généralement une complexité en considérant la pire instance possible parmi toutes les instances de taille n, c'est-à-dire celle demandant le plus de calcul. d'où la définition suivante :

Définition 1.7. Soient P un probléme et A un algorithme qui résout P. Notons $I_{(P,n)} = \{x/x$ est une instance de P et $|x| = n\}$ et φ_A est une application qui à toute instance x fait associer le temps d'exécution de A sur x . La fonction de complexité (la complexité) de l'algorithme A est une application c_A définie sur l'ensemble des entiers naturels par :

$$c_A(n) = max_{I_{(P,n)}} \varphi(n).$$

$c_A(n)$ est le nombre maximum d'opérations élémentaires pris par A sur les instances de taille n.

Dans la suite, nous nous intéresserons qu'à la complexité temporelle des problèmes étudiés.
Dans le calcul de la complexité en temps d'un algorithme, c'est pas le nombre exacte d'opérations effectuées par ce dernier qui importe, mais c'est la nature de la fonction de complexité produite par l'algorithme et qui s'exprime en fonction de n, la taille de l'instance ; ainsi que son comportement asymptotique, c'est à dire quand n tend vers l'infini (on se demande souvent de savoir si cette fonction de complexité est une constante, un logarithme, un polynôme....). Habituellement, cette complexité est exprimée à l'aide de la notation "grand O". Sa définition est la suivante.

Définition 1.8. Soient f et g deux fonctions $f; g : N \to R_+$. On dit que $f = O(g)$ (on dit aussi f est un grand O de g) lorsqu'il existe un entier n_0 et une constante réelle c tel que pour tout $n \geq n_0$, $f(n) \leq cg(n)$.

Intuitivement, cela signifie que g devient plus grande que f à partir d'un certain entier n_0, à une constante multiplicative près.

Exemple 1.5. : Soit $f(n) = 6n^4 - 2n^3 + 5$. Pour $n_0 = 1$ et pour tout $n \geq n_0$, on a

$$6n^4 - 2n^3 + 5 \leq 6n^4 + 2n^4 + 5n^4 = 13n^4$$

Donc, en prenant $c = 13$, on a $f = O(n^4)$. Autrement dit, à un facteur constant près, $f(n)$ ne croît pas plus rapidement que n^4. Il est facile de voir qu'un polynôme $P(n)$ de degré k est toujours un $O(n^k)$.

1.7.3 Classe de problèmes NP

La classe **NP** regroupe les problèmes de décision pour lesquels une solution proposée peut être vérifiée en temps polynomial (par un algorithme déterministe).

Exemple 1.6. Prenons le problème du *Travelling Salesman Problem* (TSP). Trouver le plus court chemin passant par tous les sommets d'un graphe peut être très difficile (un problème NP-complet), mais si une solution est donnée (par exemple, un itinéraire spécifique), il est facile de vérifier si cette solution est correcte en calculant la longueur du chemin et en comparant cela à la longueur minimale possible. Vérifier une solution proposée peut être effectué en temps polynomial.

1.7.4 Classe de problème P

La classe de complexité **P** regroupe tous les problèmes de décision pour lesquels il existe un algorithme déterministe dont le temps d'exécution, dans le pire des cas, est polynomial par rapport à la taille de l'entrée.

Formellement, un problème est dans **P** s'il peut être résolu par un algorithme dont la complexité temporelle est $O(n^k)$, où :

— n représente la taille de l'entrée,

— k est une constante (dépendant de l'algorithme).

En pratique, les problèmes de **P** sont considérés comme "faciles", car ils admettent des solutions efficaces. Un objectif courant en algorithmique est de trouver des algorithmes optimaux, c'est-à-dire minimisant l'exposant k pour un problème donné – ce qui constitue souvent un défi majeur.

Exemple 1.7. Prenons le problème du tri d'un tableau d'entiers. Un algorithme courant comme le tri par fusion (merge sort) a une complexité temporelle de $O(n \log n)$, ce qui est un exemple typique d'un problème dans la classe **P**, car sa complexité est polynomialement bornée.

1.7.5 Relation entre les classes P et NP

On a l'inclusion évidente suivante

$$\mathbf{P} \subseteq \mathbf{NP}$$

Tout problème résoluble en temps polynomial (classe **P**) admet une vérification en temps polynomial, donc appartient aussi à la classe **NP**. Ce qui pose problème depuis longtemps c'est de savoir s'il y a égalité entre ces deux classes. Jusqu'à présent, personne n'a su répondre à cette question. Cependant, la communauté scientifique pense fortement que cette inclusion est stricte, par conséquent, on a posé l'hypothèse que

$$\mathbf{P} \subset \mathbf{NP}$$

Celle-ci implique l'existence d'une classe de problèmes, incluse dans NP, et qui contient les problèmes de décision les plus difficiles.

1.7.6 Classe de problèmes NP-complet

La classe **NP-complèt** regroupe un ensemble de problèmes de décision qui sont les plus difficiles à résoudre dans la classe NP. Un problème est dit **NP-complet** s'il satisfait les deux conditions suivantes :

1. **Il est dans NP** : Cela signifie qu'une solution proposée pour le problème peut être vérifiée en temps polynomial.
2. **Il est au moins aussi difficile que tous les autres problèmes dans NP** : Si un algorithme efficace (en temps polynomial) existe pour résoudre un problème NP-complet, alors tous les autres problèmes de NP peuvent également être résolus en temps polynomial.

Les problèmes NP-complets sont des problèmes pour lesquels aucune solution en temps polynomial n'est connue, et il est conjecturé que ces problèmes ne peuvent pas être résolus en temps polynomial, bien que cela n'ait pas encore été prouvé (le célèbre problème $P \neq NP$).

Exemple 1.8. Un exemple classique de problème NP-complet est le *problème du voyageur de commerce* (*Travelling Salesman Problem*, TSP).

Description : Dans ce problème, un vendeur doit visiter un ensemble de villes et revenir à sa ville de départ en parcourant chaque ville exactement une fois, tout en minimisant la distance totale parcourue. Trouver la solution optimale est un problème NP-complet, car bien que vérifier une solution donnée soit faisable en temps polynomial (en calculant la distance totale et en vérifiant qu'elle est minimale), trouver la solution optimale nécessite de vérifier toutes les permutations possibles des villes, ce qui est extrêmement difficile et nécessite un temps exponentiel pour les grandes instances du problème.

Chapitre 2

État de l'art sur le problème de coloration des graphes

La coloration des graphes est l'un des problèmes centraux de la théorie des graphes et de l'optimisation combinatoire. Elle occupe une place importante aussi bien d'un point de vue théorique, en raison des nombreux résultats et conjectures qui y sont associés, que d'un point de vue pratique, grâce à ses applications dans divers domaines tels que la planification, l'allocation de ressources, ou encore les télécommunications. Dans ce chapitre, nous présentons un état de l'art couvrant la coloration des sommets et celle des arêtes.

En 1852, le Britannique **Francis Guthrie** souleva une question en apparence simple : est-il toujours possible de colorier n'importe quelle carte géographique avec seulement quatre couleurs, tout en évitant que deux pays voisins aient la même couleur ? Cette conjecture, connue sous le nom de **problème des quatre couleurs**, ne fut résolue que plus d'un siècle plus tard. Il a fallu attendre 1976 pour que deux chercheurs américains, **Kenneth Appel** et **Wolfgang Haken** de l'Université de l'**Illinois**, apportent enfin une réponse affirmative. Leur démonstration, complexe et novatrice, marqua un tournant dans l'histoire des mathématiques. Entre-temps, de nombreux mathématiciens avaient tenté de prouver cette conjecture, contribuant ainsi au développement de nouvelles théories et outils. L'une des avancées majeures fut la reformulation du problème en termes de théorie des graphes : une carte géographique peut être représentée par un graphe, où chaque pays devient un sommet et chaque frontière commune une arête reliant deux sommets. Cette approche a permis d'aborder le problème sous un angle différent, ouvrant la voie à sa résolution finale. Ainsi, ce qui semblait être une simple question de coloriage a mobilisé les esprits pendant plus de 120 ans, illustrant combien certains défis mathématiques, même élémentaires en apparence, peuvent receler une profonde complexité.

2.1 Nombre chromatique d'un graphe

Étant donné un graphe $G = (V, E)$, une **coloration des sommets** de G est une application $f : V \to C$, où C désigne l'ensemble des couleurs. Évidemment, attribuer des couleurs aux sommets d'un graphe n'aurait que peu d'intérêt si celles-ci n'étaient pas soumises à au moins une contrainte. Par exemple, si l'on reprend le cas des cartes géographiques, deux sommets adjacents doivent être de couleurs différentes. Ce type de coloration, dite **coloration propre des sommets** (proper coloring). Ainsi, Une **coloration propre des sommets** d'un graphe G est une attribution de couleurs à ses sommets telle que deux sommets adjacents reçoivent des couleurs distinctes. Formellement

Définition 2.1. Étant donné un graphe $G = (V, E)$, une **coloration propre des sommets** de G est une application $f : V \to C$, telle que

$$\forall u, v \in V, uv \in E \Leftrightarrow f(u) \neq f(v)$$

Il est toujours possible de colorer un graphe « proprement » : il suffit d'attribuer une couleur différente à chaque sommet. Mais cette solution n'est clairement pas à privilégier, dans la mesure où pour la majorité des problèmes les couleurs symbolisent des ressources que l'on cherche justement à économiser. Ainsi, le problème de la coloration d'un graphe consiste-t-il à déterminer le nombre minimum de couleurs pour le colorer :

Problème de la coloration de graphe (Vertex Coloring)

Instance : Un graphe $G = (V, E)$;
Question : Déterminer une coloration de G utilisant un nombre minimum de couleurs.

Ce nombre est appelé **nombre chromatique** (chromatic number) de G, et est noté $\chi(G)$. Ainsi, le nombre chromatique de G est le plus petit nombre de couleurs nécessaires pour colorier correctement les sommets de G.

Un graphe $G = (V, E)$ est dit *k-coloriable* (ou admet une *k-coloration*) s'il est possible d'assigner une couleur à chaque sommet de V parmi k couleurs différentes, de sorte que deux sommets reliés par une arête soient toujours de couleurs distinctes.

Cela revient à dire qu'il existe une fonction :

$$f : V \to \{1, 2, \ldots, k\}$$

telle que :

$$\forall (u, v) \in E, \quad f(u) \neq f(v).$$

Notons que les sommets d'une même couleur étant deux à deux non adjacents, ils forment un stable. Autrement dit :

Proposition 2.1. *Un graphe est k-colorable si et seulement s'il est possible de partitionner son ensemble de sommets en k stables.*

Il est important de noter que le problème de décider si un graphe est k-coloriable, pour un entier $k \geq 3$, est un problème ***NP-complet*** *[22]. En conséquence, la détermination exacte du nombre chromatique $\chi(G)$ est un problème difficile d'un point de vue algorithmique, c'est-à-dire il n'existe à ce jour aucun algorithme polynomial connu pour le résoudre sur les graphes généraux.*

Ce résultat s'inscrit dans la théorie de la complexité des problèmes combinatoires, où la coloration est devenue un problème emblématique. De nombreux travaux ultérieurs ont étudié des cas particuliers (par exemple les graphes parfaits [23] ou les graphes planaires via le théorème des quatre couleurs [24]), ainsi que des algorithmes approchés et heuristiques [25].

Propriétés

Voici quelques propriétés fondamentales du nombre chromatique $\chi(G)$, accompagnées de leurs démonstrations :

P_1 *- Pour tout graphe G à n sommets, on a la majoration suivante :*

$$\chi(G) \leq n$$

Démonstration. Le nombre chromatique $\chi(G)$ correspond au nombre minimal de couleurs nécessaires pour colorier les sommets de G de telle sorte que deux sommets adjacents aient toujours des couleurs différentes. Dans le pire des cas, où chaque sommet est relié à tous les autres (graphe complet K_n), on utilise n couleurs distinctes. Ainsi, pour tout graphe à n sommets, il suffit de n couleurs au maximum, ce qui donne :

$$\chi(G) \leq n$$

□

P_2 *- Si G est un graphe complet K_n, alors :*

$$\chi(K_n) = n$$

Démonstration. Dans un graphe complet K_n, chaque sommet est adjacent à tous les autres. Ainsi, aucune couleur ne peut être partagée entre deux sommets. Il faut donc n couleurs différentes, soit :

$$\chi(K_n) = n$$

□

P_3 - *Si le degré maximal de G est $\Delta(G) = d$, alors :*

$$\chi(G) \leq d + 1$$

Démonstration. Cette propriété peut être démontrée à l'aide de l'algorithme glouton. On parcourt les sommets de G un par un, et on assigne à chaque sommet la plus petite couleur disponible qui n'est pas utilisée par ses voisins déjà colorés. Un sommet a au plus d voisins, donc au plus d couleurs interdites. Il reste toujours une couleur disponible parmi $d + 1$, ce qui garantit que :

$$\chi(G) \leq \Delta(G) + 1$$

□

P_4 - *Si G est un graphe planaire, alors :*

$$\chi(G) \leq 4$$

Démonstration. C'est le célèbre *théorème des quatre couleurs*, démontré par Appel et Haken en 1976 à l'aide de l'ordinateur. Il affirme que tout graphe planaire peut être colorié avec au plus 4 couleurs de manière à ce que deux sommets adjacents aient toujours des couleurs différentes. On a donc :

$$\chi(G) \leq 4$$

□

2.2 Quelques algorithmes d'approximation du problème de coloration

La coloration des graphes est relativement simple lorsque le nombre de sommets est faible, mais elle devient rapidement complexe à mesurer que ce nombre augmente.

Pour résoudre ce problème, divers algorithmes ont été conçus. Ceux que nous allons présenter (parmi d'autres existants) permettent d'obtenir une coloration satisfaisante, c'est-à-dire utilisant un nombre de couleurs raisonnablement limité.

2.2.1 Algorithme glouton (greedy algorithm)

*Une première méthode pour colorier un graphe consiste à parcourir ses sommets un par un et à leur attribuer une couleur, en veillant à ce que deux sommets adjacents ne partagent jamais la même couleur. C'est ce que l'on appelle l'**algorithme***

***de coloration séquentielle**. Il s'agit d'un algorithme dit* glouton*, car il effectue un choix localement optimal — celui de la "meilleure" couleur disponible à chaque étape — dans l'espoir d'obtenir une solution globalement satisfaisante.*

*Par exemple, pour chaque sommet, on peut lui attribuer la plus petite couleur qui n'est pas déjà utilisée par ses voisins. Cette méthode garantit une **coloration propre**, c'est-à-dire une coloration correcte respectant les contraintes, mais pas nécessairement optimale en termes de nombre de couleurs utilisées. Le résultat dépend fortement de l'ordre dans lequel les sommets sont colorés.*

Il est important de noter que, pour tout graphe, il existe toujours un ordre des sommets permettant à l'algorithme glouton de produire une coloration optimale. En effet, il suffirait de connaître une telle coloration optimale à l'avance et de numéroter les sommets selon l'ordre croissant de leurs couleurs. Toutefois, cela crée un paradoxe, puisque déterminer une telle numérotation suppose déjà de connaître la solution optimale, ce qui est précisément ce que l'on cherche à obtenir.

*Dans ce qui suit, on considère un graphe $\Gamma = (V, E)$ à colorier, et on note n (respectivement m) l'ordre (respectivement la taille) de Γ. Les sommets du graphe sont alors supposés être numérotés de 0 à $n-1$, c'est-à-dire $V = \{0, 1, \ldots, n-1\}$. Les numérotations ou les ordres considérés seront donc des **permutations** de cet ensemble.*

*Par ailleurs, sauf indication contraire, lorsqu'il sera question de la complexité d'un algorithme, on entendra par défaut la **complexité temporelle**.*

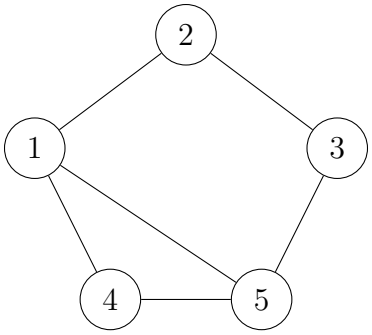
Coloration séquentielle

Entrée : *Un graphe quelconque $G = (V, E)$, une permutation des sommets $\sigma : \{0, \ldots, n-1\} \to V$*
Sortie : *Une coloration propre $c : V \to \mathbb{N}$.*

Algorithm 1 Coloration séquentielle

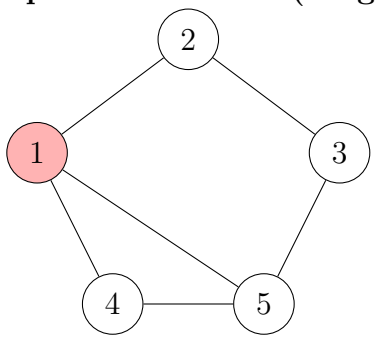
1: $n \leftarrow |V|$
2: $Couleur \leftarrow$ Tableau(Taille : n, Défaut : -1)
3: **for** $i = 0 \rightarrow n - 1$ **do**
4: $\quad x \leftarrow \sigma(i)$
5: $\quad$ # Recherche de la plus petite couleur non utilisée dans $N(x)$ #
6: $\quad Libre \leftarrow$ Tableau(Taille : n, Défaut : vrai)
7: $\quad$ **for** $y \in N(x)$ **do**
8: $\quad\quad$ **if** $Couleur(y) \neq -1$ **then**
9: $\quad\quad\quad$ **6.** $Libre(Couleur(y)) \leftarrow$ faux
10: $\quad\quad$ **end if**
11: $\quad$ **end for**
12: $\quad index \leftarrow 0$
13: $\quad$ **while** $Libre(index) =$ faux **do**
14: $\quad\quad index \leftarrow index + 1$
15: $\quad$ **end while**
16: $\quad$ # Affecter la couleur donnée par $index$ à x #
17: $\quad Couleur(x) \leftarrow index$
18: **end for**
19: **Renvoyer** $Couleur$

Exemple 2.1. **Graphe initial (non coloré)**

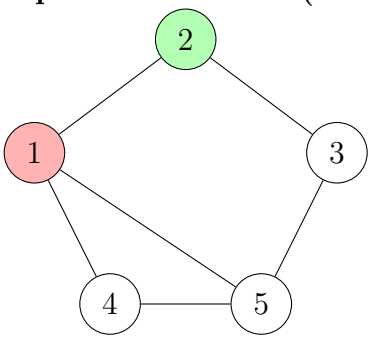


Étapes de la coloration

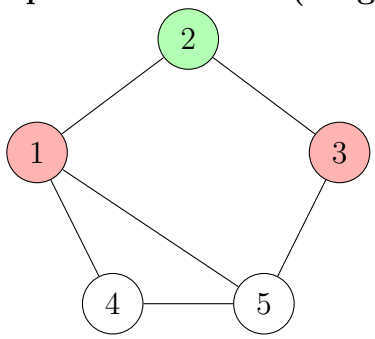
Étape 1 : sommet 1 (rouge)



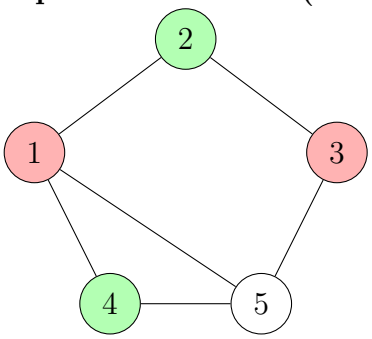
Étape 2 : sommet 2 (vert)



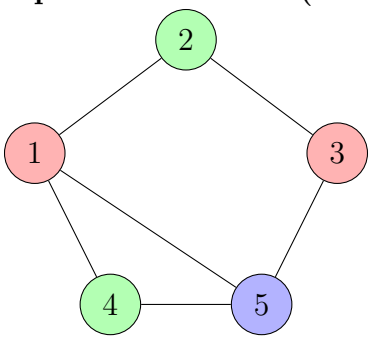
Étape 3 : sommet 3 (rouge)



Étape 4 : sommet 4 (vert)



Étape 5 : sommet 5 (bleu)



Graphe final coloré

Le graphe est correctement coloré avec **3 couleurs** : rouge, vert et bleu. Aucun sommet adjacent ne partage la même couleur.

2.2.2 Algorithme de Welsh & Powell

L'algorithme de Welsh & Powell repose sur le principe de la coloration séquentielle, mais avec un ordre des sommets soigneusement choisi, dans le but d'obtenir une coloration correcte tout en limitant le nombre de couleurs utilisées. L'idée est de colorier en priorité les sommets les plus contraignants, c'est-à-dire ceux qui ont le plus grand nombre de voisins.

Concrètement, cet algorithme consiste à parcourir les sommets du graphe dans l'ordre décroissant de leur degré (nombre de voisins). Les sommets à fort degré étant plus difficiles à colorier, on commence par eux pour maximiser les chances d'utiliser un nombre réduit de couleurs.

Bien que l'algorithme de Welsh & Powell soit couramment utilisé et permet généralement d'obtenir une coloration assez efficace, il ne garantit pas pour autant d'utiliser le nombre minimum de couleurs (appelé nombre chromatique). On parle donc d'une heuristique, car l'algorithme fournit une solution approchée, non optimale dans tous les cas.

Algorithm 2 Algorithme de Welsh & Powell pour la coloration des sommets

Require: Un graphe $G = (V, E)$ avec $n = |V|$ sommets.
Ensure: Une coloration propre des sommets de G.

```
Ordonner les sommets v_1, v_2, ..., v_n par degrés décroissants.
Initialiser tous les sommets comme non colorés.
Poser l'indice de couleur c ← 1.
while il existe des sommets non colorés do
    Sélectionner le premier sommet non coloré dans l'ordre et lui attribuer la couleur c.
    for chaque sommet v_j restant non coloré (dans l'ordre) do
        if v_j n'est adjacent à aucun sommet déjà coloré avec c then
            Attribuer la couleur c à v_j.
        end if
    end for
    Incrémenter l'indice de couleur : c ← c + 1.
end while
```

2.2.3 Algorithme de DSATUR

***DSATUR** est un algorithme de coloration de graphes proposé en 1979 par **Daniel Brélaz**, à l'école polytechnique fédérale de Lausanne (EPFL) [16]. Il améliore la stratégie de coloration séquentielle en tenant compte non seulement du nombre de voisins d'un sommet, mais aussi de la diversité des couleurs déjà présentes dans son voisinage.*

L'idée centrale de ***DSATUR*** *est la suivante : un sommet est considéré comme difficile à colorier non seulement s'il a de nombreux voisins, mais surtout si ses voisins sont déjà colorés avec plusieurs couleurs différentes. On introduit alors une notion appelée* ***degré de saturation*** *qui mesure cette contrainte : le degré de saturation d'un sommet v à un moment donné de l'algorithme est donné par son degré, si aucun voisin de v n'est colorié ; ou par le nombre de couleurs différentes déjà utilisées parmi ses voisins.*

Algorithm 3 Algorithme DSATUR (Brélaz, 1979)

Require: Un graphe $G = (V, E)$ avec $n = |V|$ sommets.
Ensure: Une coloration propre des sommets de G.

Pour tout sommet $v \in V$, marquer v comme non coloré.
Calculer le degré de chaque sommet.
Sélectionner un sommet v de degré maximum et lui attribuer la première couleur.
while il existe des sommets non colorés **do**
    Pour chaque sommet non coloré u, calculer son degré de saturation (nombre de couleurs distinctes utilisées dans son voisinage).
    Choisir le sommet u ayant le degré de saturation maximum.
    if plusieurs sommets ont le même degré de saturation **then**
        Choisir parmi eux celui de degré maximum.
    end if
    Attribuer à u la plus petite couleur possible qui n'apparaît pas dans son voisinage.
end while

Chapitre 3

Coloration des sommets d'un graphe triangulé

*Nous allons, dans ce chapitre, étudier la classe des graphes **triangulés**, laquelle est contenue dans la classe des graphes parfaits. Après avoir rappelé l'importance de cette famille, nous mettrons en évidence son inclusion dans les graphes parfaits.*

Les graphes triangulés constituent en effet l'une des classes les plus fondamentales en théorie des graphes. Ils apparaissent naturellement dans de nombreux contextes pratiques et se distinguent par leurs remarquables propriétés algorithmiques. Grâce à ces propriétés, un grand nombre de problèmes appartenant à la classe NP-complets admettent des solutions polynomiales lorsqu'ils sont restreints à cette famille, en particulier le problème de la coloration des sommets.

3.1 Graphes parfaits

Introduit dans les années 60 par shannon, la notion de graphe parfait occupe une place très importante en théorie des graphes. De nombreuses classes sont en effet incluses dans la classe des graphes parfaits : citons par exemple les graphes bipartis, les graphes de comparabilités les graphes d'intervalles les graphes triangulés .... L'importance des graphes parfait est prouvée par leurs applications dans de nombreux domaines concrets.

Définition 3.1. Soit $G = (V, E)$ un graphe

1. Une **clique** est un ensemble de sommets C qui sont deux à deux adjacents, c'est à dire
$$\forall x, y \in C, xy \in E$$

Le nombre de sommets dans une clique maximum est noté $\omega(G)$.

2. Un **stable** est un ensemble de sommets S qui sont deux à deux non adjacents, c'est à dire
$$\forall x, y \in S, xy \notin E$$
Le nombre de sommets dans un stable maximum est appelé **nombre de stabilité** de G, il est noté $\alpha(G)$.
3. Une partition des sommets en clique est un ensemble de cliques qui sont deux à deux disjointes et dont la réunion est V. Le plus petit nombre de cliques qui partitionnent V est noté $\theta(G)$.

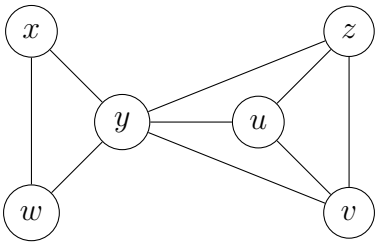


FIGURE 3.1 – clique maximale et maximum.

Dans la figure 3.1, $C_1 = \{x, y, w\}$ est une clique maximale d'ordre 3 ; elle n'est pas maximum. La clique $C_2 = \{y, z, u, v\}$ d'ordre 4 est maximum.

Remarque 3.1. Un stable maximum dans G est une clique maximum dans $\overline{G}$, ainsi
$$\omega(G) = \alpha(\overline{G})$$

Dans la figure 3.1 $S = \{x, u\}$ est un stable de cardinalité 2. Il est maximum, car il n'y a aucun autre stable avec un nombre de sommets plus grand. Ainsi, $\alpha(G) = 2$.

Remarque 3.2. Il est facile de voir que pour tout graphe simple, on a les inégalités suivantes

1. $\omega(G) \leq \chi(G)$;
2. $\alpha(G) \leq \theta(G)$.

Définition 3.2. Un graphe G est parfait si et seulement si, pour tout sous-graphe induit H de G,
$$\omega(H) = \chi(H) \tag{3.1}$$

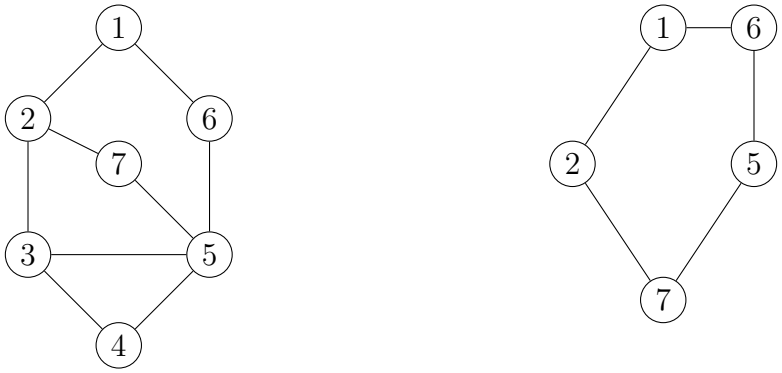


FIGURE 3.2 – Un graphe G à gauche ; un trou C_5 à droite

Théorème 3.1. *Pour tout graphe simple, on a*

$$Si : \alpha(G) = \theta(G) \Leftrightarrow \omega(G) = \chi(G) \tag{3.2}$$

Remarque 3.3. D'après le théorème 3.1, un graphe G est parfait si et seulement si , pour tout sous-graphe induit H de G

$$\theta(H) = \alpha(H) \tag{3.3}$$

Définition 3.3. On appelle **corde** dans un graphe toute arête reliant deux sommets non consécutifs d'une chaine ou d'un cycle. Un **trou** est un cycle sans corde. On désigne par C_k un trou ayant k sommets.

Dans la figure 3.2 ci-dessous, on à un graphe G à gauche et un trou C_5 à droite. L'arête e=35 est une corde dans le cycle $\mu = (1, 2, 3, 4, 5, 6)$, donc μ n'est pas un trou.

Remarque 3.4. Notons qu'une corde dans un cycle est une arête n'appartenant pas à celui-ci, et un cycle peut avoir plusieurs cordre. Le plus petit trou dans un graphe est un C_4.

Définition 3.4. On appelle **Anti-trou.** le complémentaire d'un trou est appelé anti-trou.

Il est à noter que C_5 est un trou et anti-trou.

Remarque 3.5. Notons que dans tout trou impair C_{2k+1}, on a

$$\omega = 2 \neq \chi = 3, \theta = k + 1 \neq \alpha = k \tag{3.4}$$

Il en résulte qu'un trou impair n'est pas un graphe parfait, et un graphe parfait ne peut pas contenir un trou impair comme un sous graphe induit.

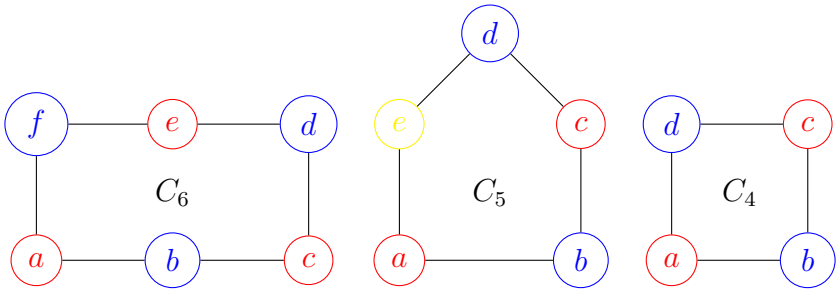


FIGURE 3.3

Théorème 3.2. *(lovasz 1972(Théorème faible))*
Un graphe G est parfait si et seulement si son complémentaire est parfait.

Il en résulte de ce théorème que les anti-trous impairs ne sont pas parfaits.

Théorème 3.3. *(Chudnovsky et al 2002(Théorème fort))*
Un graphe G est parfait si et seulement s'il ne contient ni trou ni anti-trou impairs.

3.2 Graphes triangulés

Définition 3.5. Tout graphe sans trou est appelé graphe **triangulé** (cordal graph).

Le graphe de la figure 3.2 n'est pas triangulé puisqu'il contient le trou C_5.

Proposition 3.4. *Tout sous graphe induit d'un graphe triangulé est triangulé. Autrement dit les graphes triangulés sont* ***héréditaires****.*

Démonstration. Il suffit de remarquer que si H est un sous graphe induit de G et si H contient un trou, alors ce même trou de H est aussi un trou de G. □

La définition suivante est utile pour montrer que les graphes triangulés sont parfaits.

Définition 3.6. On appelle $2K_2$ une somme disjointe de deux cliques d'ordre deux (voir figure 3.4).
Comme le montre la figure 3.4, on peut voir facilement que le complémentaire d'un $2K_2$ est un C_4, c'est à dire $\overline{2K_2} = C_4$.
On remarque aussi que $2K_2$ est l'anti-trou de C_4.

FIGURE 3.4 – Le graphe $2K_2$ et son complémentaire.

Remarque 3.6. La figure 3.5 montre le trou C_6 et son anti-trou $\overline{C_6}$. Remarquons que ce trou contient un $2K_2$ et son anti-trou contient un C_4. Ce résultat peut être généralisé facilement comme suit : tout trou d'ordre supérieur à 5 contient un $2K_2$, et donc tout anti-trou de longueur supérieure à 5 contient un C_4.

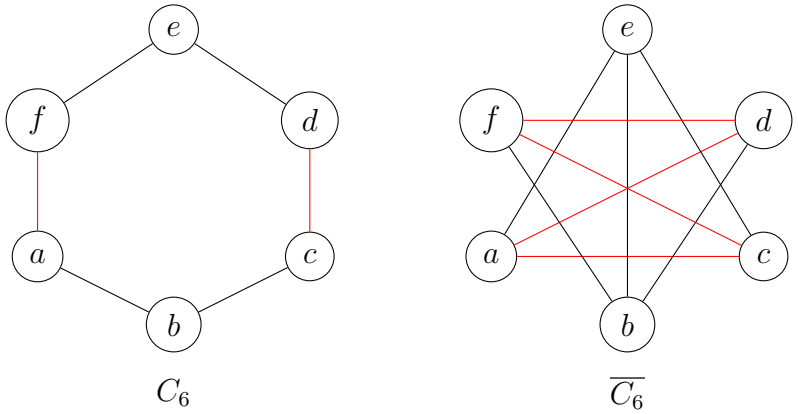


FIGURE 3.5

Théorème 3.5. *Les graphes triangulés sont parfaits.*

Démonstration. Par définition, un graphe triangulé est sans trou, donc il est sans trou impair. Puisque tout anti-trou de longueur 5 est un trou et puisque tout anti-trou d'ordre supérieur à 5 contient un C_4 (voir la remarque 3.6), alors un graphe triangulé ne peut pas contenir un anti-trou impair. Par conséquent, les graphes triangulés sont parfaits. □

Soit G un graphe connexe et soit S un ensemble de sommets.
On dit que S est un **séparateur** de G si le sous graphe de G obtenu par la

suppression de S n'est pas connexe, c'est à dire le sous graphe induit $G[V \setminus S]$ n'est pas connexe. Pour deux sommets x, y, on dit que S est un x, y-séparateur si les sommets x et y appartiennent à deux composantes connexes distinctes dans $G[V \setminus S]$. Un séparateur S est minimal si pour tout $s \in S$, $S \setminus \{s\}$ n'est pas un séparateur.

Lemme 3.6. *Soit S un séparateur dans G et $C_1, C_2 ... C_k$, les composantes connexes obtenues après la suppression de S. Alors*
S est minimal si et seulement si pour tout $x \in S$ et pour tout $i = 1...k$, on a $N(x) \cap C_i \neq \emptyset$.

Démonstration. Soit $x \in S$. sans perte de généralité, supposons que $N(x) \cap C_1 = \emptyset$. Le sous graphe $G[V - S]$ n'est pas connexe. Puisque $N(x) \cap C_1 = \emptyset$, alors $G[(V - S) \cup \{x\}] = G[V - (S - \{x\})]$ n'est pas connexe aussi. Il en résulte que $S - \{x\}$ est un séparateur, mais ceci est en contradiction avec le fait que S est minimal. Par conséquent, $N(x) \cap C_i$ n'est pas vide pour tout i.

Réciproquement, soit $x \in S$. Puisque, pour tout i, $N(x) \cap C_i \neq \emptyset$, alors $S - \{x\}$ ne peut pas être un séparateur et par conséquent S est minimal. □

Proposition 3.7. *Dans un graphe triangulé, tout séparateur minimal est une clique.*

Démonstration. Soient S un séparateur minimal et C_1, C_2 deux composantes connexes dans $G[V - S]$. Supposons que S n'est pas une clique, donc il existe $x, y \in S$ tels que $xy \notin E$, et d'après le lemme 3.6, il existe $x_1, y_1 \in C_1$ et $x_2, y_2 \in C_2$ tels que xx_1, xx_2, yy_1 et yy_2 sont des arêtes. D'autre par, puisque C_1, C_2 sont connexes, alors il existe deux chaines μ_1 et μ_2 qui connecte x_1 à y_1 dans C_1 et x_2 à y_2 dans C_2 respectivement. Il en résulte que $\mu = (x, x_1, \mu_1, y_1, y, y_2, \mu_2, x_2, x_1$ est un cycle qui manque d'une corde qui joint x et y. Il en résulte que G n'est pas triangulé. Par conséquent, si G est triangulé, alors tout séparateur minimal est une clique. □

Définition 3.7. On dit qu'un sommet x est simplicial si l'ensemble de ses voisins $N(x)$ induit une clique. Ainsi, $N[x] = N(x) \cup \{x\}$ est une clique maximale.

Proposition 3.8. *Tout graphe triangulé possède au moins un sommet simplicial. Si ce graphe n'est pas complet, alors il contient au moins deux sommets simpliciaux non adjacents.*

Définition 3.8. Pour un graphe G, soit $\sigma = (x_1, x_2 \dots x_n)$ un ordre sur les sommets de G. On dit que σ est un ordre d'élimination simplicial ou un ordre d'élimination parfait (o.e.p) si pour $i = 1....n$, le sommet x_i est simplicial dans le sous graphe induit par $\{x_i, x_{i+1}...x_n\}$

Théorème 3.9. *Un graphe G est triangulé si et seulement si il admet un ordre d'élimination parfait.*

Démonstration. Puisque tout graphe triangulé possède un sommet bisimplicial et que tout sous graphe induit d'un graphe triangulé est triangulé, alors l'application récursive de la proposition 3.8 à un graphe triangulé permet de construire un o.e.p : soit x_1 un sommet simplicial de G. $G_2 = G[\{x_2...x_n\}]$ est aussi triangulé, donc il existe x_2, un sommets simplicial de G_2. $G_3 = G[\{x_3...x_n\}]$ est triangulé, donc il existe x_3, un sommet simplicial dans G_3....

Par définition, si le graphe G n'est pas triangulé, alors il contient un trous comme un sous graphe induit. Il en résulte qu'aucun sommet de ce trou ne peut être supprimé simplicialement. □

Théorème 3.10. *[17] Soit G un graphe ordonné par Lex-BFS $\sigma = (v_1, v_2, ..., v_n)$. Alors, G est triangulé si et seulement si σ est un ordre d'élimination simplicial.*

Le dernier théorème conduit à un algorithme de reconnaissance d'un graphe triangulé en un temps polynomiale. En effet, pour savoir si un graphe donné est triangulé, on lui applique le parcours Lex-BFS et verifier si l'ordre produit sur les sommets est un ordre d'élimination simpliciale.

Exemple 3.1. Dans la figure 3.6, on à un graphe G d'ordre 5, et à gauche, on a le même graphe auquel on a appliqué le parcours Lex-BFS. On peut verifier facilement que l'ordre $\sigma = (12345)$ est un ordre d'élimination simpliciale. Par conséquent, le graphe G est triangulé.

3.3 Coloration d'un graphe triangulé

Comme il a été signalé plus haut, le problème de coloration des sommets est un problème NP-complet quand le graphe d'entré est quelconque. Cependant, pour les graphes triangulés (ou graphes chordaux), ce problème admet une solution polynomiale grâce à leurs bonnes propriétés structurelles. Pour colorer un graphe triangulé, on utilise le principe suivant :

1. On applique le parcours Lex-BFS pour un graphe triangulé pour calculer un ordre d'élimination simpliciale (O.E.S) $\sigma = (v_1, v_2, ..., v_n)$.

FIGURE 3.6 – Un graphe G à gauche et le même graphe à droite ordonné par Lex-BFS

2. On parcourir les sommets dans l'ordre inverse du O.E.S $(v_n, \ldots, v_1)$ en attribuant à chaque sommet v_i, la plus petite couleur qui n'est pas utilisée par ses voisins déjà colorés.

Algorithm 4 Coloration optimale d'un graphe triangulé

Entrée : Un graphe triangulé $G = (V, E)$ et un O.E.S $(v_1, \ldots, v_n)$;
Sortie : Une coloration optimale de G.

1: **Pour tout** sommet $v \in V$ **faire** couleur$(v) \leftarrow$ Non assignée
2: **for** $i \leftarrow n$ **downto** 1 **do**
3: $C \leftarrow$ ensemble des couleurs utilisées par les voisins déjà colorés de v_i
4: couleur$(v_i) \leftarrow$ plus petit entier non présent dans C
5: **end for**
6: **return** $\{ couleur(v) \mid v \in V \}$

3.4 Application au problème de la fibre optique

Dans les réseaux de fibre optique partagés, plusieurs entreprises peuvent avoir besoin d'accéder à une bande passante commune sur des plages horaires qui se chevauchent. Pour éviter les interférences et garantir une qualité de service optimale, on cherche à attribuer à chaque entreprise un **canal** de communication distinct pour chaque période d'utilisation. Ce problème peut être modélisé par un **graphe d'intervalles**, où chaque sommet représente une entreprise, et chaque arête signale un chevauchement temporel.

Définition des intervalles

Voici un exemple de 12 entreprises avec leurs plages horaires d'utilisation (en heures) de la bande passante sur une journée :

Sommet	**Entreprise**	**Intervalle d'utilisation (heures)**
v_1	E_1	[1,3]
v_2	E_2	[2,5]
v_3	E_3	[3,6]
v_4	E_4	[5,8]
v_5	E_5	[4,7]
v_6	E_6	[7,9]
v_7	E_7	[5,9]
v_8	E_8	[4,6]
v_9	E_9	[5,8]
v_{10}	E_{10}	[6,8]
v_{11}	E_{11}	[7,10]
v_{12}	E_{12}	[8,10]

TABLE 3.1 – Tableau des intervalles d'utilisation de la fibre optique par les 12 entreprises.

Construction du graphe d'intervalles

On connecte deux entreprises si leurs intervalles d'utilisation se chevauchent.

Numérotation LexBFS

En appliquant l'algorithme **LexBFS** au graphe de la figure 3.7 , on obtient l'ordre de numérotation suivant :

$$v_1, v_2, v_3, v_5, v_4, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}$$

Cet ordre respecte la propriété des graphes d'intervalles : à chaque étape, le sommet choisi est celui ayant l'étiquette lexicographiquement maximale parmi les sommets restants.

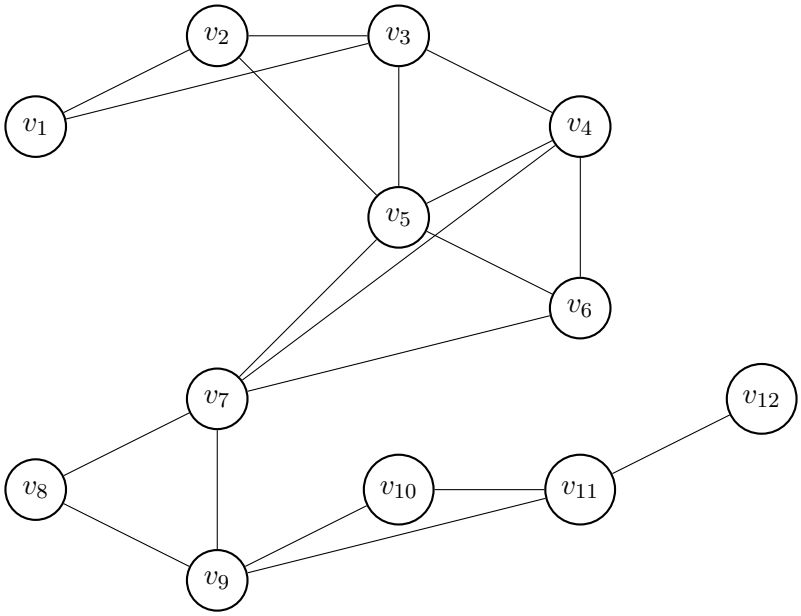


FIGURE 3.7 – Modélisation graphique du problème.

Reconnaissance du graphe

On souhaite vérifier si le graphe étudié est triangulé. Rappelons qu'un graphe est **triangulé** s'il ne contient aucun cycle induit de longueur supérieure ou égale à 4 sans corde. Une manière algorithmique de le vérifier consiste à appliquer l'algorithme **LexBFS** : si celui-ci fournit un **ordre d'élimination parfait** (PEO, *Perfect Elimination Ordering*), alors le graphe est triangulé.

Application de LexBFS

En appliquant l'algorithme LexBFS sur notre graphe (Figure 3.7), on obtient la numérotation hiérarchique suivante :

$$v_1, v_2, v_3, v_5, v_4, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}.$$

Vérification de l'ordre d'élimination parfait

Un ordre est dit PEO si, pour tout sommet v_i, les voisins de v_i qui apparaissent après lui dans l'ordre forment une clique. Vérifions pour quelques sommets clés :

- Pour v_1 : ses voisins ultérieurs $\{v_2\}$ forment trivialement une clique.
- Pour v_3 : ses voisins ultérieurs $\{v_4, v_5\}$ sont reliés entre eux, donc c'est une clique.

— Pour v_5 : ses voisins ultérieurs $\{v_4, v_6, v_7\}$ sont tous reliés (avec l'arête v_4-v_7 ajoutée), donc c'est une clique.
— Pour v_9 : ses voisins ultérieurs $\{v_{10}, v_{11}\}$ sont connectés, donc c'est une clique.

L'ensemble des vérifications montre que chaque sommet satisfait la condition d'élimination parfaite.

Conclusion :

Ainsi, l'ordre donné par LexBFS constitue un **ordre d'élimination parfait**. On conclut donc que le graphe étudié est bien un **graphe triangulé**.

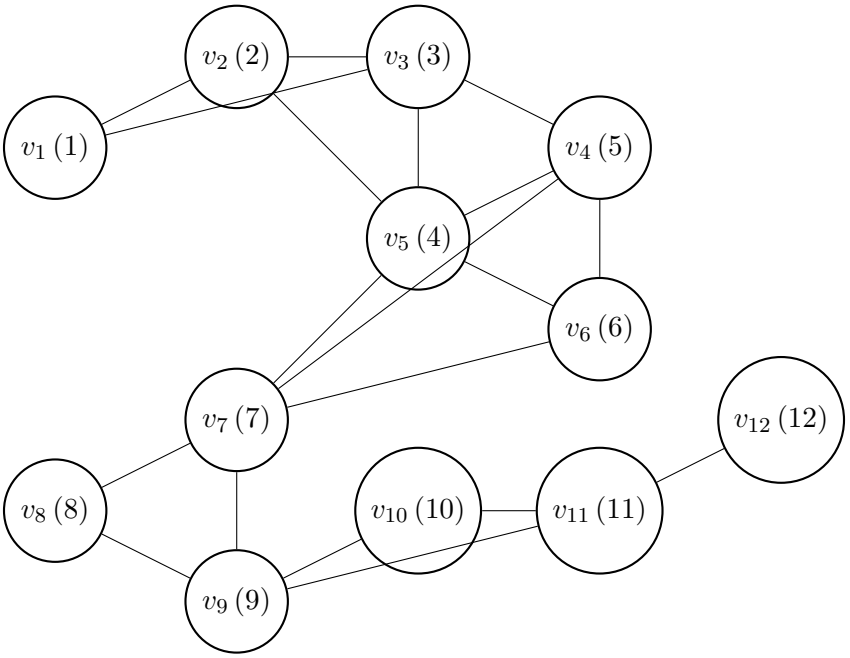


FIGURE 3.8 – Numérotation obtenue par LexBFS (ordre 1..12) sur le graphe donné.

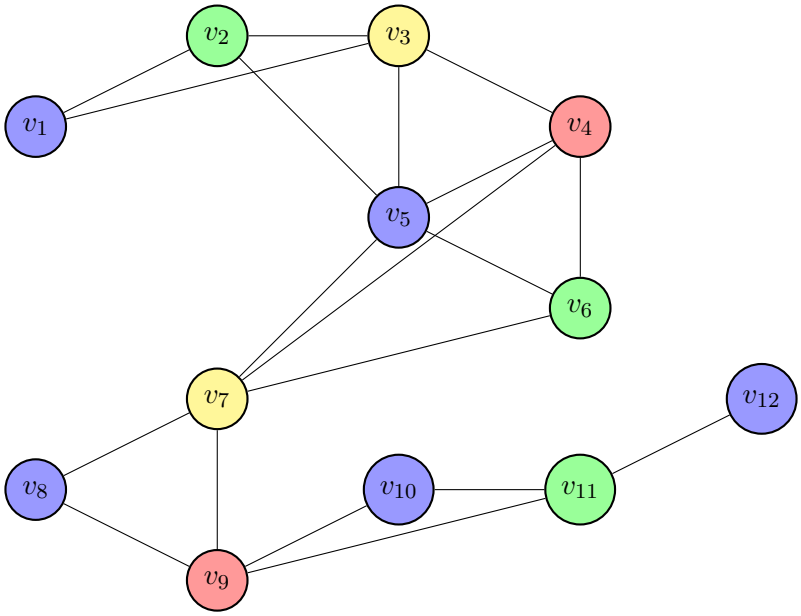


FIGURE 3.9 – Coloration gloutonne du graphe avec 4 couleurs.

L'exemple étudié illustre clairement la puissance des graphes d'intervalles dans la gestion de la bande passante partagée en fibre optique. En modélisant les périodes d'utilisation des entreprises par des intervalles, nous avons pu construire le graphe d'intervalles associé et appliquer l'algorithme **LexBFS**. Ce dernier a fourni un ordre d'élimination parfait (PEO), confirmant que le graphe est bien triangulé.

Par ailleurs, l'application de la **coloration gloutonne** a montré que seulement quatre canaux distincts suffisent pour éviter tout chevauchement et garantir une utilisation optimale des ressources. Cette approche démontre ainsi l'intérêt pratique des graphes d'intervalles pour résoudre des problèmes réels liés à la planification et à la gestion efficace des réseaux de communication.

Chapitre 4

Coloration des arêtes d'un graphe biparti

La coloration des arêtes d'un graphe est une variante naturelle et importante du problème de coloration des sommets. Rappelons que la coloration des sommets consiste à attribuer des couleurs aux sommets de manière à ce que deux sommets adjacents reçoivent des couleurs différentes. La coloration des arêtes s'intéresse à l'attribution des couleurs aux arêtes de telle sorte que deux arêtes incidentes à un même sommet portent des couleurs distinctes.

Les applications pratiques du problème de coloration des arêtes sont nombreuses. Par exemple, dans la planification d'emplois du temps, chaque arête peut représenter un cours associé à une salle et un enseignant, et la coloration correspond alors à une affectation d'horaires sans conflit. De même, dans les réseaux de communication, les arêtes représentent des canaux de transmission qui doivent utiliser des fréquences distinctes lorsqu'ils partagent un nœud commun, afin d'éviter les interférences. Dans l'attribution de tâches : certaines chaînes de production ou de logistique, une machine ne peut exécuter qu'une tâche à la fois. Une bonne coloration des arêtes permet d'éviter les conflits d'utilisation des ressources. Dans le problèmes de transport : dans les réseaux ferroviaires ou aériens, on doit organiser les trajets ou les pistes d'atterrissage de manière à ce que deux lignes incidentes à une gare ou un aéroport ne soient pas planifiées au même moment.... . Ces exemples montrent que la coloration des arêtes n'est pas uniquement un sujet théorique, mais qu'elle possède une importance majeure dans des domaines aussi variés que l'éducation, les réseaux, la logistique et les transports.

Dans ce chapitre, nous allons présenter les notions fondamentales de la coloration des arêtes, en insistant sur son lien avec les couplages, puis nous nous focaliserons sur le cas particulier des graphes bipartis où l'indice chromatique peut être déterminé efficacement. Enfin, nous illustrerons ces concepts par l'application au problème de l'emploi du temps.

4.0.1 Définitions et propriétés générales

Soit $G = (V, E)$ un graphe simple et soit k un entier strictement positif. Une **coloration des arêtes** du graphe G est une application $c : E \to \mathbb{N}$ telle que $c(a_1) \neq c((a_2)$ chaque fois que les arêtes a_1 , a_2 sont incidentes. Ainsi, une **coloration des arêtes** d'un graphe G consiste à attribuer une couleur à chaque arête de telle sorte que deux arêtes adjacentes (c'est-à-dire partageant un sommet) reçoivent des couleurs différentes.

On appelle **k-arête-coloration** une coloration des arêtes avec k couleurs, i.e. $|c(E)| = k$. S'il n'y a aucune ambiguïté on dira simplement k-coloration. Le plus petit entier k tel que le graphe G admette une k-arête-coloration est appelé **indice chromatique**, et on le notera $\chi'(G)$. Un graphe G admettant une k-arête-coloration est dit **k-arête-colorable** (ou simplement k-colorable). Le graphe de la figure ci-dessous admet une 4-coloration des arêtes, donc il est 4-colorable.

Exemple 4.1. Soit le graphe suivant :

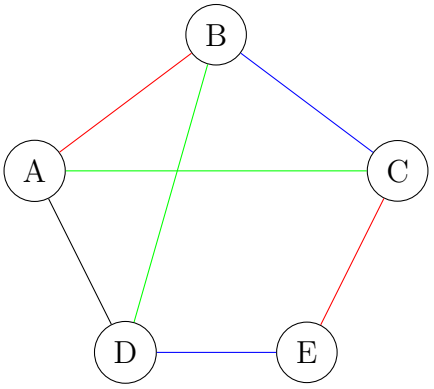


FIGURE 4.1 – Graphe 4-arête-colorable

Remarque 4.1. Remarquons que tout graphe G ayant m arêtes est m-colorable. Ceci montre que $\chi'(G) \leq m$. Par ailleurs, si x est un sommet de degré maximum dans G, c'est-à-dire $d(x) = \Delta(G)$, alors les $\Delta(G)$ arêtes de G ayant le sommet x pour extrémité doivent toutes être de couleurs distinctes. Ceci montre que la coloration des arêtes de G nécessite au moins $\Delta(G)$ couleurs, c'est-à-dire $\Delta(G) \leq \chi'(G)$. D'où l'encadrement suivant de l'indice chromatique.

$$\Delta \leq \chi' \leq m$$

En 1964, **Vizing** a trouvé un encadrement optimale. En effet, il a montré que pour tout graphe simple G, le paramètre $\chi'(G)$ peut prendre que deux valeurs. plus précisément, on a

Théorème 4.1 (Vizing, 1964)**.** *Pour tout graphe G de degré maximum $\Delta(G)$, on a :*

$$\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$$

Ainsi, le théorème de Vizing partitionne l'ensemble des graphes selon leurs indice chromatique en deux classes :

— **Première classe** : les graphes pour lesquels $\chi'(G) = \Delta(G)$;
— **Deuxième classe** : les graphes pour lesquels $\chi'(G) = \Delta(G) + 1$.

Démonstration. On démontre les deux inégalités séparément.

1. $\chi'(G) \geq \Delta(G)$: Cette inégalité est immédiate. En effet, le sommet de degré maximum $\Delta(G)$ est incident à $\Delta(G)$ arêtes, qui doivent toutes avoir des couleurs différentes dans une coloration propre. Il faut donc au moins $\Delta(G)$ couleurs.
2. $\chi'(G) \leq \Delta(G) + 1$: On construit progressivement une coloration des arêtes du graphe G, en utilisant au plus $\Delta(G) + 1$ couleurs.

Soit une arête $e = uv$ non encore colorée. Notons :

— $C(u)$: l'ensemble des couleurs utilisées par les arêtes incidentes à u ;
— $\bar{C}(u) = \{1, \ldots, \Delta + 1\} \setminus C(u)$: les couleurs disponibles à u ;
— De même pour $C(v)$ et $\bar{C}(v)$.

Si une couleur est disponible à la fois à u et à v, on l'utilise pour colorier uv. Sinon, on procède à une recoloration locale via des *chaînes de Kempe* :

1. Choisir une couleur $\alpha \in \bar{C}(u)$, et une couleur $\beta \in \bar{C}(v)$;
2. Construire une chaîne alternée entre les couleurs α et β, reliant u ou v à d'autres sommets ;
3. Échanger les couleurs le long de cette chaîne pour libérer une couleur disponible à l'un des deux sommets ;
4. Colorier finalement l'arête uv.

Ce procédé fonctionne pour chaque arête et garantit qu'on n'utilise jamais plus de $\Delta + 1$ couleurs. D'où :

$$\chi'(G) \leq \Delta(G) + 1.$$

□

Remarque 4.2. Le théorème est valable uniquement pour les graphes simples. Dans le cas des graphes multiarêtes, on a une généralisation par Vizing avec une borne supérieure plus élevée.

L'indice chromatique d'un graphe G quelconque vaut toujours Δ ou $\Delta + 1$. Cependant, comme il est montré dans [[8]], le problème de décider entre ces deux valeurs est NP-complet. Il existe quand même quelques classes de graphes pour

lesquels on peut savoir facillement si ellee sont dans la première ou la deuxième classe.

Proposition 4.2 ([1])**.** *Soit $n \geq 2$.*

1. *Si n est impair, K_n est de classe 2 : $\chi'((K_n) = n = \Delta((K_n) + 1$.*
2. *Si n est pair, K_n est de classe 1 : $\chi'((K_n) = n - 1 = \Delta(Kn)$.*

Exemple 4.2.
— Le graphe K_3 (triangle) a $\Delta = 2$ et $\chi'(K_3) = 3 = \Delta + 1$. C'est donc un graphe de **deuxième classe**.
— Tout graphe biparti est de **première classe**, d'après le théorème de König : $\chi'(G) = \Delta(G)$.
— Le graphe K_5 (complet à 5 sommets) a $\Delta = 4$ et $\chi'(K_5) = 5 = \Delta + 1$: il est de deuxième classe.

Le théorème de Channon suivant offre une estimation sur l'indice chromatique des multigraphes, c'est-à-dire des graphes dans lesquels plusieurs arêtes peuvent relier les mêmes sommets.

Théorème 4.3. *Shannon (1949) Soit M un multigraphe de degré maximum $\Delta(M)$. Alors :*

$$\chi'(M) \leq \left\lfloor \frac{3}{2}\Delta(M) \right\rfloor$$

Démonstration. On considère un multigraphe M de degré maximum Δ. Chaque sommet est donc incident à au plus Δ arêtes. Comme il peut y avoir plusieurs arêtes entre deux sommets, on ne peut pas appliquer directement le théorème de Vizing.

L'idée consiste à partitionner l'ensemble des arêtes de M en un certain nombre de classes d'arêtes telles que, dans chaque classe, chaque sommet a un degré au plus 2. Un tel sous-graphe est formé uniquement de cycles, chemins ou composantes isolées, et peut donc être colorié à l'aide de 3 couleurs au maximum.

Shannon a démontré qu'il est toujours possible de partitionner l'ensemble des arêtes de M en au plus $\lfloor \frac{3}{2}\Delta \rfloor$ telles classes. En coloriant chaque classe avec une couleur différente, on obtient une coloration propre des arêtes de M, ce qui prouve que :

$$\chi'(M) \leq \left\lfloor \frac{3}{2}\Delta \right\rfloor .$$

□

Remarque 4.3. Cette borne n'est pas optimale pour tous les multigraphes, mais elle est valable de manière générale. Il existe des cas où $\chi'(M) < \lfloor \frac{3}{2}\Delta(M) \rfloor$, mais jamais de cas où elle est dépassée.

Exemple 4.3. Soient deux sommets u et v reliés par m arêtes parallèles. Le degré de chaque sommet est m, donc $\Delta(M) = m$ et l'indice chromatique est :

$$\chi'(M) = m.$$

En revanche, la borne de Shannon donne :

$$\left\lfloor \frac{3}{2} \cdot m \right\rfloor.$$

Par exemple, si $m = 4$, alors $\chi'(M) = 4$ et $\left\lfloor \frac{3}{2} \cdot 4 \right\rfloor = 6$. L'inégalité est donc vérifiée.

4.1 Couplage dans les graphes bipartis

La notion de couplage dans un graphe est l'une des grandes notions de la théorie des graphes. Elle est une notion générale, mais nous nous restreignons ici au cas des graphes bipartis, et nous verrons comment l'utiliser pour le calcul d'une coloration optimale efficacement dans cette classe de graphes. Rappelons qu'un graphe $G = (V, E)$ est dit **biparti** si son ensemble de sommets V peut être partitionné en deux ensembles X et Y tels que chaque arête relie un sommet de X à un sommet de Y. On écrit alors $G = (X, Y, E)$.

Définition 4.1 (Couplage)**.** Un **couplage** est un ensemble d'arêtes $M \subseteq E$ deux à deux disjointes, c'est-à-dire sans sommets en communs. Un couplage M est **maximum** si sa cardinalité est maximale, c'est-à-dire, si M' est un autre couplage, alors $|M'| \leq |M|$. Un couplage M est dit **maximal** si on ne peut pas l'inclure dans un autre couplage, c'est-à-dire, pour toute arête $e \notin M$, $\quad e \cup M$ n'est pas un couplage.

Dans la figure 4.2, l'ensemble des arêtes $M = \{12, 46\}$ est un couplage. Ce couplage ne peut pas être agrandir, donc il est maximal. Cependant, ce même couplage n'est pas maximum, l'ensemble $M' = \{13, 26, 45\}$ est un autre couplage ayant 3 éléments.

Un sommet $v \in V = X \cup Y$ est **saturé** par le couplage M s'il est incident à une arête de M. S'il n'est incident à aucune arête de M, il est dit **libre**.

Définition 4.2. Une chaîne **alternée** relativement au couplage M, ou chaîne M -alternée, est une chaîne élémentaire μ dont les arêtes sont alternativement dans M et dans $E \setminus M$. Une chaîne M-alternée est dite **augmentante** pour M, ou M-augmentante, si ses extrémités sont libres.

Le théorème suivant donne une condition nécessaire et suffisante pour qu'un couplage donné soit maximum. Ce même théorème est à la base d'un algorithme efficace pour calculer un couplage maximum dans un graphe biparti.

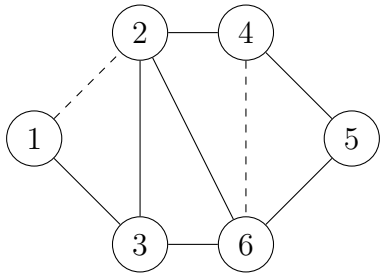


FIGURE 4.2 – Couplage

Théorème 4.4 (Berge, 1957)**.** *Un couplage M est maximum si et seulement s'il n'existe pas de chaîne augmentante relativement à M.*

L'algorithme suivant basé sur le théorème de Berge calcule efficacement un couplage maximum dans un graphe biparti. Pour tout sommet v, $Pred(v)$ est le sommet voisin de v (i.e. $Pred(v) \in N(v)$), marqué avant v et qui a permis le marquage de v, donc $Pred(v)$ et v sont consécutifs dans une certaine chaine M-alternée.

Algorithm 5 Algorithme de marquage pour la recherche d'un couplage maximum

Entrée : Un graphe biparti $G = (X, Y, E)$ et un couplage M.
Sortie : M est maximum ssi il n'existe pas de chaîne M-augmentante.

1: $L \leftarrow$ sous-ensemble de X constitué des sommets non saturés par M
2: **for all** $v \in X \cup Y$ **do**
3: marquer$(v) \leftarrow$ Faux
4: $pred(v) \leftarrow \emptyset$
5: **end for**
6: **while** $L \neq \emptyset$ **do**
7: Choisir un sommet $w \in L$; $L \leftarrow L \setminus \{w\}$
8: marquer$(w) \leftarrow$ Vrai
9: $F \leftarrow \{w\}$ ▷ file d'attente
10: **while** $F \neq \emptyset$ **do**
11: Prendre le premier sommet x de F ; $F \leftarrow F \setminus \{x\}$
12: **for all** $y \in N(x)$ tel que y non marqué **do**
13: **if** y est saturé par une arête $yz \in M$ **then**
14: $F \leftarrow F \cup \{z\}$
15: $pred(y) \leftarrow x$
16: $pred(z) \leftarrow y$
17: **else**
18: $pred(y) \leftarrow x$
19: effectuer un transfert sur la chaîne μ reliant w à y
20: $M \leftarrow M \triangle \mu$
21: **return** M
22: **end if**
23: **end for**
24: **end while**
25: **end while**
26: **return** M

Preuve de la correctitude

(1) Correction partielle. Si l'algorithme retourne après avoir reconstruit une chaîne μ et effectué $M \leftarrow M \,\Delta\, \mu$, alors $|M|$ augmente de 1. En effet, par définition d'une chaîne M-augmentante, ses arêtes alternent entre $E \setminus M$ et M et commencent/finissent hors de M. L'opération Δ (symétrique) enlève de M les arêtes de μ qui y figuraient et y ajoute celles qui n'y figuraient pas ; comme les extrémités sont libres, on gagne exactement une arête dans M.

(2) Correction négative. Supposons que l'algorithme termine sans avoir trouvé de chaîne augmentante. Par construction, tout sommet accessible depuis L par une chaîne alternée a été marqué (propagation BFS : arêtes hors M vers Y non marqué ; puis, si y est saturé, on marque aussi son partenaire z via l'arête de M). S'il existait une chaîne M-augmentante μ reliant un sommet libre de X à un sommet libre de Y, ses sommets seraient tous marqués par ces règles et l'algorithme l'aurait détectée au moment d'atteindre le sommet libre de Y — contradiction. Il n'existe donc pas de chaîne M-augmentante.

(3) Maximalité ⇒ optimalité (Berge). Par le théorème de Berge, l'absence de chaîne M-augmentante équivaut à l'optimalité de M. Ainsi, si l'algorithme ne trouve aucune chaîne, M est maximum ; s'il en trouve une, il augmente strictement $|M|$.

Terminaison

Chaque sommet est marqué au plus une fois ; chaque arête (x, y) est examinée au plus une fois par la boucle interne. Les structures F (file) et L (sources libres) s'épuisent en un nombre fini d'opérations. L'algorithme termine.

Complexité

Notons $n = |X| + |Y|$ et $m = |E|$.

- **Une recherche (à partir d'un $w \in L$)** coûte $O(n + m)$: initialisation $O(n)$; exploration en largeur visitant chaque arête au plus une fois ; reconstruction d'une chaîne μ via `pred` en $O(n)$.
- **Nombre d'augmentations** $\leq \lfloor n/2 \rfloor$ (chaque augmentation ajoute une arête au couplage).

Ainsi, pour construire un couplage maximum en itérant cette recherche :

$$O\big(n \cdot (n + m)\big)$$

soit $O(n^3)$ dans le pire cas dense ($m = \Theta(n^2)$). Pour améliorer, on peut utiliser **Hopcroft–Karp** (1973) qui cherche des *familles* de chaînes augmentantes de plus courte longueur par couches BFS/DFS et atteint $O(m\sqrt{n})$.

Dans un graphe biparti tel que $|X| \leq |Y|$, il est important de savoir s'il existe un couplage M saturant tous les sommets de X, c'est-à-dire un couplage M avec $|M| = |X|$. Le théorème de Hall ci-dessous donne une condition nécessaire et suffisante pour qu'un tel couplage existe. Ce même théorème, bien sûr, peut aussi être utilisé pour montrer l'existence d'un couplage parfait quand $|X| = |Y|$.

Théorème 4.5 (Théorème de Hall)**.** *Soit $G = (X, Y, E)$ un graphe biparti. Il existe un couplage qui sature tous les sommets de X si et seulement si pour tout sous-ensemble $S \subseteq X$,*

$$|N(S)| \geq |S|,$$

où $N(S)$ désigne l'ensemble des voisins de S dans Y.

Définition 4.3 (Couplage parfait)**.** Un couplage M est dit **parfait** s'il sature tous les sommets du graphe, c'est-à-dire si $|M| = \frac{|V|}{2}$.

Notons que tout couplage parfait est maximum. De plus, pour qu'un graphe G admettent un couplage parfait, il est nécessaire que $|X| = |Y|$. Le théorème suivant, constituant un cas particulier du théorème de Hall, donne une condition nécessaire est suffisante pour l'existence d'un couplage parfait dans un graphe biparti.

Théorème 4.6. *(Frobenius)[[11]] Soit $G = (X, Y, E)$ un graphe biparti. Alors G a un couplage parfait si et seulement si $|X| = |Y|$ et $|N(S) \geq |S|$ pour tout $S \subset X$.*

Le théorème suivant est une conséquence des deux premiers théorèmes. Il affirme que dans un graphe biparti dont tous les sommets ont un même degré, il existe toujours un couplage parfait.

Théorème 4.7 (Kőnig)**.** *Tout graphe biparti k-régulier (chaque sommet est de degré k) admet un couplage parfait.*

4.2 Coloration optimale d'un graphe biparti

Le lien entre couplage et coloration des arêtes est essentiel : en effet, une coloration des arêtes correspond à une partition de l'ensemble des arêtes en couplages. Chaque couleur définit un couplage, puisque deux arêtes de même couleur ne peuvent être adjacentes. Ainsi, colorier les arêtes revient à recouvrir le graphe par un nombre minimal de couplages.

Un résultat classique dû à König (1916) établit les graphes bipartis appartiennent à la première classe.

Théorème 4.8. *Pour tout graphe biparti, on a*

$$\chi'(G) = \Delta(G).$$

Ce théorème permet de résoudre efficacement le problème de coloration des arêtes dans les graphes bipartis, car il existe des algorithmes polynomiaux pour trouver une telle coloration. Ces algorithmes reposent en grande partie sur la décomposition du graphe en couplages parfaits ou maximaux.

Exemple 4.4. Considérons le graphe biparti complet $K_{3,3}$. Chaque sommet ayant un degré 3, on a $\Delta(K_{3,3}) = 3$. Par le théorème de König, $\chi'(K_{3,3}) = 3$. Il est donc possible de colorier les arêtes de $K_{3,3}$ avec exactement trois couleurs, en partitionnant les 9 arêtes en trois couplages parfaits.

Remarque 4.4. D'après le théorème de König ci-dessous, tout graphe biparti régulier admet un couplage parfait. Remarquons que si G est un graphe k-régulier et M un couplage parfait, alors le graphe partiel obtenu par suppression des arêtes de M est un graphe $(k-1)-$ régulier et donc, d'après König, ce graphe admet un couplage parfait. On peut continuer ce processus et après k itération on aura k couplages parfaits $M_1, M_2, ..., M_k$.

4.2.1 Coloration optimale d'un graphe régulier

Soit $G = (X, Y, E)$ un graphe biparti Δ-régulier. On a donc :
— $|X| = |Y|$,
— Pour tout sommet $v \in X \cup Y$, $d(v) = \Delta(G)$.
Pour colorer les arêtes de G, on procède comme suit :

1. Poser $G_1 = G$.
2. Appliquer l'algorithme 5 afin de calculer un couplage parfait M_1 dans G_1.
3. Attribuer la couleur c_1 aux arêtes de M_1.
4. Construire $G_2 = G_1 - M_1$, obtenu en supprimant les arêtes de M_1 dans G_1.
5. Répéter les étapes précédentes sur chaque graphe G_i jusqu'à épuisement de toutes les arêtes.

On obtient ainsi une suite de graphes partiels

$$G_1 = G,\ G_2,\ \ldots,\ G_\Delta,$$

et une suite de couplages parfaits

$$M_1,\ M_2,\ \ldots,\ M_\Delta.$$

Il est à noter que pour tout $i = 1, \ldots, \Delta$:
— $|M_i| = |X| = |Y| = \dfrac{|E|}{\Delta}$,
— Le dernier graphe partiel coïncide avec le dernier couplage, c'est-à-dire $G_\Delta = M_\Delta$.
La procédure de coloration s'arrête alors, et les arêtes de G sont colorées avec exactement Δ couleurs distinctes : $c_1, c_2, \ldots, c_\Delta$.

Exemple 4.5.

4.2.2 Coloration d'un biparti non régulier

L'observation suivante est une conséquence du théorème de König sur l'indice chromatique d'un graphe biparti.

Observation 1. *Si G est un graphe biparti, alors pour tout sous-graphe H de G vérifiant $\Delta(H) = \Delta(G)$, on a*

$$\chi'(H) = \chi'(G) = \Delta(G)$$

Le théorème suivant montre que tout graphe biparti G de degré maximum k peut être prolongé en un graphe biparti k-régulier H contenant G. Plus précisément

Théorème 4.9 ([13])**.** *Soit $G = (X, Y, E)$ un graphe biparti simple. Alors il existe un graphe biparti simple $H = (X', Y', E')$ tel que*
- *$X \subseteq X', \quad Y \subseteq Y'$ et $E \subset E'$ (autrement dit H contient G),*
- *H est régulier,*
- *$\Delta(H) = \Delta(H)$*

(G) Graphe biparti non régulier (H) Graphe régularisé

FIGURE 4.3 – (G) Graphe biparti non régulier et (H) sa régularisation en ajoutant des sommetset des arêtes fictifs.

Exemple 4.6.

Méthode de calcul pratique

Soit $G = (X, Y, E)$ un graphe biparti simple quelconque de degré max Δ. On construit un graphe biparti $G' = (X' \cup Y', E')$ **Δ-régulier** contenant G comme sous-graphe. On applique ensuite la procédure de décomposition en couplages parfaits (cas régulier citée plus haut) sur G' : pour $i = 1...\Delta$, trouver un couplage parfait M'_i, colorier M'_i avec la couleur c_i, retirer M'_i. Pour chaque i poser $M_i := M'_i \cap E$; attribuer la couleur c_i aux arêtes de M_i. Les arêtes fictives sont ignorées au résultat final.

Exemple 4.7. Considérons le graphe biparti G représenté dans la figure 4.4-(G). Ce graphe n'est pas régulier puisque les degrés de ses sommets ne sont pas tous égaux. Pour appliquer l'algorithme de partition en couplages parfaits, nous construisons un graphe biparti régulier H en ajoutant un sommet fictif x_4 et les arêtes fictives x_4y_1 et x_4y_4. Ainsi, H est un graphe biparti 2-régulier contenant G (voir figure 4.4-(H)).

En appliquant l'algorithme de marquage à H, on obtient une partition des arêtes en deux couplages parfaits :

$$M_1' = \{x_1y_1,\ x_2y_2,\ x_3y_3,\ x_4y_4\}, \qquad M_2' = \{x_1y_2,\ x_2y_3,\ x_3y_4,\ x_4y_1\}.$$

En restreignant ces couplages aux arêtes de G, on obtient une partition en deux couplages :

$$M_1 = \{x_1y_1,\ x_2y_2,\ x_3y_3\}, \qquad M_2 = \{x_1y_2,\ x_2y_3,\ x_3y_4\}.$$

Ainsi, les arêtes de G peuvent être colorées avec deux couleurs : la première pour

(G) Graphe biparti non régulier (H) Graphe régularisé

FIGURE 4.4 – (G) Graphe biparti non régulier et (H) sa régularisation par ajout d'un sommet et d'arêtes fictifs.

M_1, et la seconde pour M_2. On en déduit que

$$\chi'(G) = 2.$$

4.3 Application : problème de l'emploi de temps

On considère 7 professeurs $P_1, \ldots, P_7$ et 6 classes $C_1, \ldots, C_6$. Chaque arête (P_i, C_j) indique que le professeur P_i doit enseigner à la classe C_j. Les salles attribuées pour chaque professeur sont données comme suit :

— $P_1 : C_1, C_3$
— $P_2 : C_2, C_4, C_5$

— $P_3 : C_1, C_2$
— $P_4 : C_3, C_5, C_6$
— $P_5 : C_2, C_6$
— $P_6 : C_4, C_5$
— $P_7 : C_1, C_4, C_6$

Problème : trouver un emploi de temps minimisant le nombre de créneaux (ou plages horaires) dans la journée ?

Pour modéliser la situation, on utilise le graphe biparti $G = (P, C, E)$ suivant :

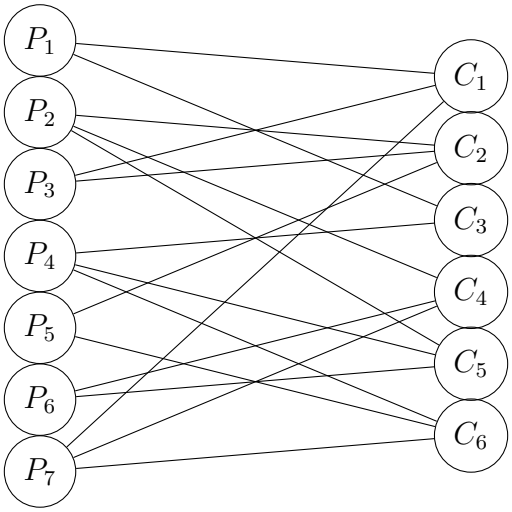


FIGURE 4.5 – Graphe biparti initial : arêtes = cours

Dans le graphe de la figure 4.5, P est l'ensemble des sommets représentant les professeurs ; C l'ensemble des sommets représentant les salles ; un couple de sommets P_i, C_j sont reliés par une arête si et seulement si le professeur P_i a une séance de cours dans la salle C_j. Ainsi, résoudre ce problème revient à colorer les arêtes du graphe G en utilisant le minimum possible de couleur. D'une manière équivalente, il s'agit de partitionner les arêtes de G en un nombre minimum de couplages. On sait que le nombre minimum de couplages est $\Delta(G)$. Remarquons que le graphe G n'est pas régulier et son degré maximum est $\Delta = 3$. Ainsi, pour résoudre ce problème, on doit d'abord régulariser le graphe G.

Régularisation

Pour appliquer la méthode de décomposition en couplages parfaits, on régularise le graphe en ajoutant :

— une classe fictive F_1 reliée à P_1, P_3, P_5 ;
— une arête supplémentaire (P_6, C_3).
On obtient ainsi un graphe 3-régulier.

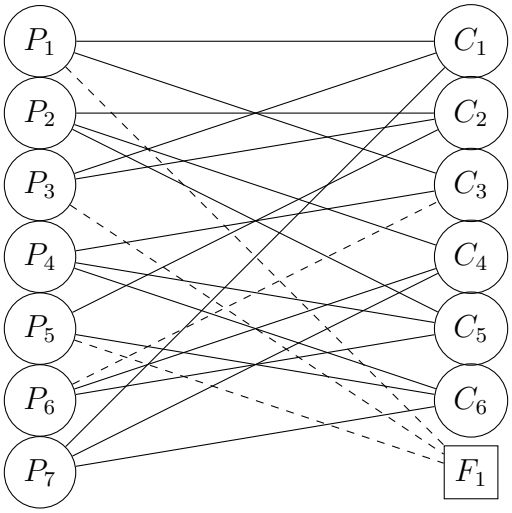
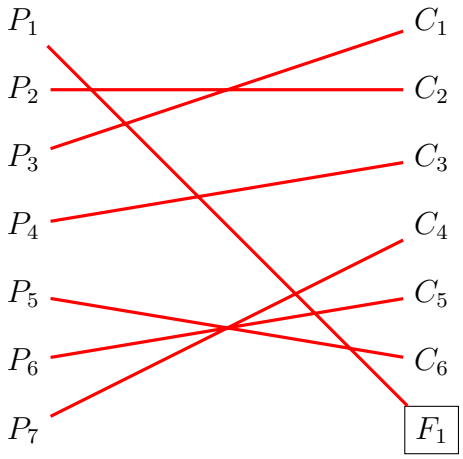


FIGURE 4.6 – Graphe régularisé (3-régulier)

3. Décomposition en 3 couplages parfaits

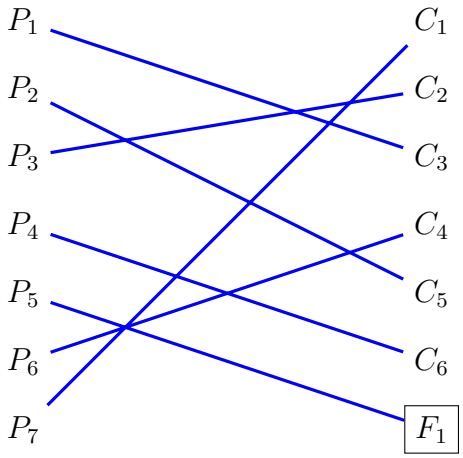
Le graphe 3-régulier admet une décomposition en 3 couplages parfaits (créneaux horaires) :

$$M_1 = \{(P_1, F_1), (P_2, C_2), (P_3, C_1), (P_4, C_3), (P_5, C_6), (P_6, C_5), (P_7, C_4)\}$$



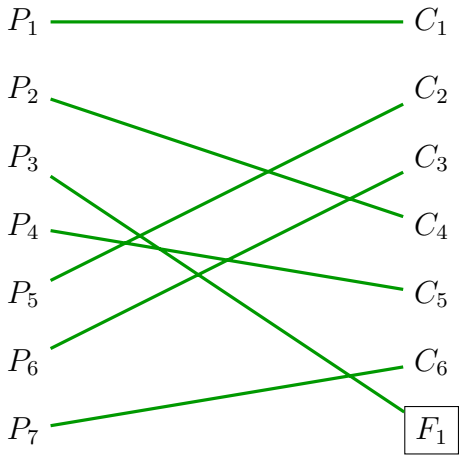
M1 (créneau 1) — rouge

$$M_2 = \{(P_1, C_3), (P_2, C_5), (P_3, C_2), (P_4, C_6), (P_5, F_1), (P_6, C_4), (P_7, C_1)\}$$



M2 (créneau 2) — bleu

$$M_3 = \{(P_1, C_1), (P_2, C_4), (P_3, F_1), (P_4, C_5), (P_5, C_2), (P_6, C_3), (P_7, C_6)\}$$



M3 (créneau 3) — vert

4. Tableau emploi du temps réel (sans F_1)

On supprime les arêtes fictives pour obtenir l'emploi du temps réel :

Créneau	**Professeur**	**Classe**
M1	P_2–C_2, P_3–C_1, P_4–C_3, P_5–C_6, P_6–C_5, P_7–C_4	
M2	P_1–C_3, P_2–C_5, P_3–C_2, P_4–C_6, P_6–C_4, P_7–C_1	
M3	P_1–C_1, P_2–C_4, P_4–C_5, P_5–C_2, P_6–C_3, P_7–C_6	

Ainsi, en 3 créneaux seulement, tous les cours sont placés sans conflit : aucun professeur n'a deux cours au même créneau, et aucune classe ne reçoit deux professeurs en même temps.

Conclusion Générale

Au terme de ce mémoire, nous avons mené une étude approfondie sur la coloration des graphes, en mobilisant à la fois les aspects théoriques, algorithmiques et applicatifs.

Dans une première partie, nous avons présenté les notions fondamentales de la théorie des graphes ainsi que les bases de la complexité algorithmique, offrant ainsi un cadre rigoureux pour aborder les différentes problématiques liées à la coloration.

Nous avons ensuite étudié les principaux algorithmes de parcours (BFS, DFS, LexBFS) ainsi que les méthodes de coloration, en insistant sur leurs propriétés, leurs performances et leurs limites.

Par la suite, nous avons exploré les graphes parfaits et certaines classes particulières, telles que les graphes triangulés et les graphes bipartis, qui offrent des résultats remarquables en lien avec la coloration. L'application de ces résultats nous a permis de mettre en évidence des solutions efficaces à des problèmes complexes de gestion et d'optimisation.

Enfin, nous avons illustré nos travaux à travers plusieurs exemples concrets issus des sciences de gestion, tels que la planification des emplois du temps, la gestion des ressources partagées ou encore l'allocation dans les réseaux de communication. Ces applications démontrent que la théorie des graphes, loin de rester une discipline abstraite, constitue un outil puissant d'aide à la décision et un levier d'optimisation dans de nombreux domaines.

Ce mémoire confirme donc la place centrale de la théorie des graphes dans les sciences appliquées contemporaines, en soulignant sa capacité à fournir des solutions élégantes, efficaces et généralisables à des problèmes réels.

En perspective, il serait intéressant d'approfondir l'étude de nouvelles classes de graphes, d'améliorer les méthodes de coloration, ou encore d'intégrer ces modèles dans des outils informatiques avancés afin de renforcer leur utilité pratique.

Nous espérons que ce travail contribue, à son échelle, à mettre en lumière la richesse théorique et l'importance applicative de la théorie des graphes, et plus particulièrement de la coloration, dans la résolution de problèmes complexes en mathématiques appliquées et en optimisation.

Bibliographie

[1] Alain B., Alain F., François H. Éléments de théorie des graphes, Springer-Verlag France, 2012.

[2] K. Appel and W. Haken. *Solution of the Four Color Map Problem. Scientific American*, vol. 237, no. 4, pp. 108–121, 1977.

[3] Bondy J.A, et Murty U. S. R., Théorie des grphes, Springer 2008.

[4] M.C. Golumbic Algorithmic Graph Theory and Perfect Graphs, Elsevier, Second Edition (2004).

[5] G. Gutin and A. P. Punnen (Eds.). *The Traveling Salesman Problem and Its Variations*. Springer, 2002.

[6] L. Lovász. *A Characterization of Perfect Graphs. Journal of Combinatorial Theory*, Series B, vol. 13, no. 2, pp. 95–98, 1972.

[7] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas. *The Strong Perfect Graph Theorem. Annals of Mathematics*, vol. 164, pp. 51–229, 2006.

[8] IAN HOLYER, THE NP-COMPLETENESS OF EDGE-COLORING, SIAM J. COMPUT. Vol. 10, No. 4, November 1981.

[9] P. Seymour. *How the Proof of the Strong Perfect Graph Conjecture Was Found. Gazette des Mathématiciens*, no. 106, pp. 7–21, 2006.

[10] Gardner, M. (1984). *The Sixth Book of Mathematical Games from Scientific American*. University of Chicago Press. pp. 92-94.

[11] Bernhard Korte and Jens Vygen, Optimisation combinatoire, Théorie et algorithmes, Springer-Verlag France 2010.

[12] Nishizeki, T. et Rahman, M.S. (2004). *Planar Graph Drawing*. World Scientific Publishing.

[13] Yun William Yu, Extending bipartite graphs to regular bipartite graphs, question sur MathStackExchange, 2012.

[14] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest et Clifford Stein. *Introduction to Algorithms*. 4e édition, MIT Press, 2022.

[15] Khadidja Jettouii, *Coloration des graphes : structures et algorithmes*, Mémoire de Master, Université Abou Bekr Belkaïd, Tlemcen, 2019.

[16] D. Brélaz, *New methods to color the vertices of a graph*, Communications of the ACM, vol. 22, no. 4, pp. 251–256, 1979.

[17] D.J. Rose, R.E. Tarjan, and G.S. Lueker : Algorithmic aspects of vertex elimination on graphs, SIAM Journal on Computing, vol. 5, p. 266-283(1976).

[18] C. Berge, *Théorie des graphes et ses applications*, Dunod, 1958.

[19] L. Lovász, M. D. Plummer, *Matching Theory*, North–Holland, 1986.

[20] J. E. Hopcroft, R. M. Karp, "An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs", *SIAM Journal on Computing*, 1973.

[21] H. W. Kuhn, "The Hungarian Method for the Assignment Problem", *Naval Research Logistics Quarterly*, 1955.

[22] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, Springer, 1972.

[23] C. Berge, *Farbung von Graphen, deren sämliche bzw. deren ungerade Kreise starr sind*, Wissenschaftliche Zeitschrift, 1961.

[24] K. Appel and W. Haken, *Every planar map is four colorable*, Bulletin of the American Mathematical Society, 1977.

[25] M. R. Garey and D. S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[26] S. Olariu. *An optimal greedy heuristic to color interval graphs.* Information Processing Letters, vol. 37, no. 1, pp. 21–25, 1991.

[27] F. F. Dragan. *LexBFS-orderings and powers of graphs.* In Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science, vol. 1665, pp. 166–179, 1999.

[28] F. F. Dragan, F. Nicolai, and A. Brandstädt. *LexBFS-orderings and powers of graphs with applications to graph powers.* Discrete Mathematics, vol. 224, no. 1-3, pp. 67–80, 2000.