Différents problèmes en théorie des graphes

Eglantine Camby*
eglantine.camby@ulb.ac.be

Résumé

L'organisation des horaires dans une école, l'allocation de fréquences dans un réseau de télécommunication ou la résolution du Sudoku sont des problèmes difficiles. Il existe un modèle mathématique permettant d'aborder ces situations de manière efficace : le graphe. Dans cet article, nous présentons brièvement les principales définitions et nous étudions plusieurs théorèmes importants. Nous illustrons cette introduction de plusieurs exemples célèbres, dont le théorème des cinq couleurs.

Sommaire

*Eglantine Camby est doctorante et Assistante au Département de Mathématique de l'Université libre de Bruxelles. Elle est titulaire d'un Master en Sciences Mathématiques de l'Université de Mons, et travaille en théorie des graphes.

1 Introduction

Au milieu du XX^{e} siècle, le célèbre physicien hongrois Eugène Wigner parle de « la déraisonnable efficacité des mathématiques dans les sciences de la nature. » En effet, la modélisation mathématique facilite la compréhension d'un problème car elle détermine un seul vocabulaire formel pour différentes situations, et elle permet de trouver une méthode de résolution automatique via un programme informatique. La modélisation mathématique peut atteindre un niveau d'abstraction permettant le développement d'une théorie précise sur le modèle, indépendante de la réalité, tout en gardant de nombreuses applications réelles. Dans cet article, nous allons aborder un modèle mathématique particulier : le graphe.

Un *graphe* est un ensemble de points, appelés *sommets*, reliés entre eux par des lignes, appelées *arêtes*. Que vous en soyez conscient ou non, vous avez déjà rencontré divers graphes au quotidien, voir figures 1 et 2. Certains graphes sont beaucoup plus complexes que d'autres. Les figures 3 et 4 marquent ce contraste.

Figure 1 — Le réseau social Facebook.

Outre les moyens de transport, nous croisons les graphes dans de nombreux contextes : les réseaux de télécommunications (internet, téléphonie, ...), les circuits éléctriques, les hiérarchies de fichiers informatiques, les bases de données relationnelles, le stockage de données, le flux de contrôle dans un programme, le codage, les multiples relations entre personnes d'un même groupe, la séquence ARN (biologie), les différentes interactions dans un écosystème (écologie), la représentation des molécules (chimie) et bien d'autres applications : organiser l'ordonnancement de tâches, les services de secours, ...

Avant de parler de problèmes spécifiques, en moins de deux pages nous allons vous munir du vocabulaire nécessaire à la compréhension de cet article. Tout d'abord, définissons formellement un graphe.

Définition 1. Un *graphe* est une paire $G = (V_G, E_G)$ formée d'un ensemble V_G de sommets et E_G d'arêtes uv de sommets u et v, c'est-à-dire $E_G \subseteq \{uv | u, v \in V_G\}$.

Nous notons $|V_G|$ le nombre de sommets et $|E_G|$ le nombre d'arêtes du graphe. Dans un graphe, la notion de connexion entre sommets, peut-être une des notions les plus importantes, se définit comme suit.

Définition 2. Deux sommets u et v sont *adjacents* ou *voisins* dans G si uv est une arête de G. Les sommets u et v sont les *extrémités* de l'arête uv.

Définition 3. Le *degré* d'un sommet v de G est le nombre d'arêtes incidentes à ce sommet, ou encore le nombre de voisins de v dans G.

Définition 4. Un graphe est *complet* si tous ses sommets sont adjacents entre eux.

Deux sommets non adjacents peuvent malgré tout avoir une certaine connexion. Dans ce cas, nous parlerons de chemin.

Définition 5. Un *chemin* dans $G = (V_G, E_G)$ de longueur n est une suite de sommets $a_1 a_2 \dots a_n a_{n+1}$ reliés entre eux par des arêtes, c'est-à-dire $a_i a_{i+1} \in E_G$, pour $i = 1, 2, \dots, n$. Un *cycle* est un chemin fermé, c'est-à-dire que $a_1 = a_{n+1}$.

FIGURE 2 — Métro bruxellois.

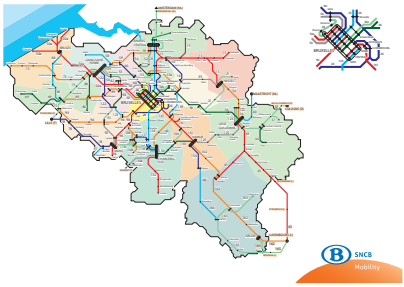


FIGURE 3 — Le réseau ferroviaire belge.

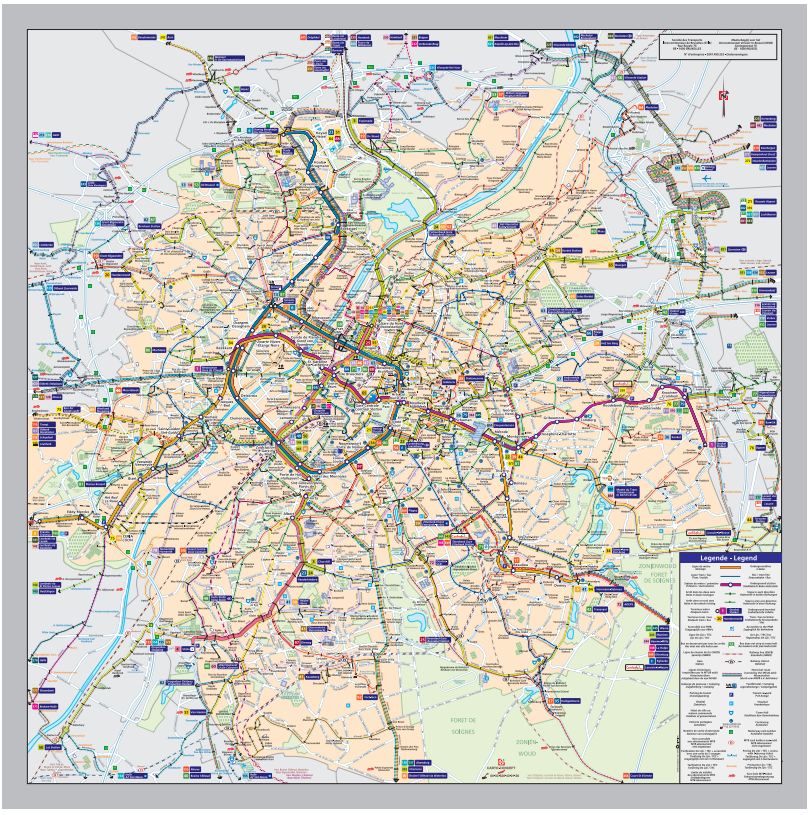
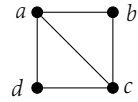


FIGURE 4 — Le réseau des transports en commun bruxellois.

Exemple 6.

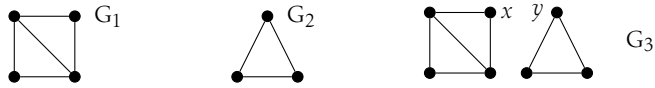
Voici un graphe $G = (V_G, E_G)$ où $V_G = \{a, b, c, d\}$ et $E_G = \{ab, ac, ad, bc, cd\}$. Les sommets a et c sont de degré 3 alors que b et d sont de degré 2. De même, a et c sont voisins alors que ce n'est pas le cas pour b et d. La suite de sommets $abcad$ forme un chemin et non un cycle de G. Par contre, $abca$ et $adcba$ sont deux cycles de G.



De plus, un graphe peut contenir une connexion entre tous ses sommets. Ceci illustre la notion de connexité.

Définition 7. Un graphe est *connexe* s'il existe un chemin dans ce graphe entre chaque paire de sommets.

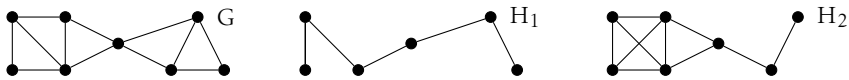
Exemples. Les deux graphes G_1 et G_2 ci-dessous sont connexes tandis que le graphe G_3 ne l'est pas, puisqu'il n'existe aucun chemin entre les sommets x et y par exemple.



Lorsque nous nous intéressons qu'à une partie d'un graphe, c'est-à-dire à certains sommets et certaines arêtes, avec des contraintes spécifiques, nous parlerons de sous-graphe.

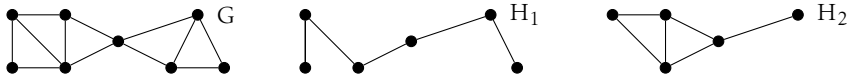
Définition 8. Un *sous-graphe* $H = (V_H, E_H)$ d'un graphe $G = (V_G, E_G)$ est le graphe G auquel des sommets et/ou des arêtes ont été enlevés, c'est-à-dire $V_H \subseteq V_G$ et $E_H \subseteq E_G$.

Exemples. Le graphe H_1 est un sous-graphe de G alors que H_2 n'en est pas un.



Définition 9. Le sous-graphe H de $G = (V_G, E_G)$ *induit* par $U \subseteq V_G$, noté G[U], est défini par $V_H = U$ et $E_H = \{xy \in E_G | x, y \in U\}$, autrement dit les arêtes de H sont celles de G dont les deux extrémités sont dans U.

Exemples. Le graphe H_1 n'est pas un sous-graphe induit de G alors que H_2 en est un. Mais H_1 et H_2 sont tous les deux des sous-graphes de G.



2 Les sept ponts de Königsberg

La naissance officielle de la théorie des graphes remonte à 1741 lorsque Euler s'est promené dans la ville de Königsberg, maintenant Kaliningrad, où deux branches du cours d'eau Pregel se rencontrent pour rejoindre la mer Baltique. Les différentes parties de la ville sont reliées par sept ponts, comme indiqué sur la figure 5. Le problème résolu par Euler [1] consiste à trouver une promenade

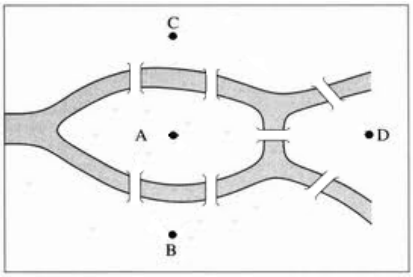


FIGURE 5 — Les sept ponts de Königsberg.

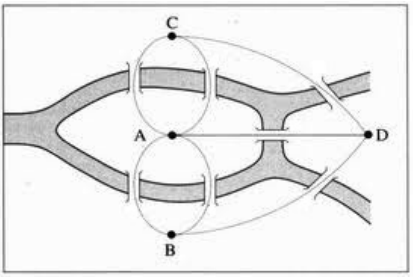


FIGURE 6 — Modélisation du problème des sept ponts de Königsberg.

partant d'un point, passant exactement une et une seule fois par chaque pont et revenant à ce point de départ.

Ce problème est à première vue banal car il suffit de donner une telle promenade pour affirmer l'existence d'une solution. Par contre, montrer qu'il n'existe pas de tel parcours reste nébuleux sans l'aide des mathématiques. Euler a prouvé qu'il est impossible d'effectuer une telle promenade, en utilisant la notion de graphe. En effet, ce modèle décrit de façon complète et très simple le problème. Un sommet est placé dans chaque partie de la ville et une arête relie deux sommets en fonction de la présence d'un pont entre ces parties, comme indiqué sur la figure 6. Euler découvre ainsi un des premiers théorèmes de la théorie des graphes.

Théorème 10. *Un graphe connexe admet un cycle passant une et une seule fois par chaque arête, si et seulement si, tous les sommets du graphe ont un degré pair.*

En mémoire à son inventeur, un tel cycle est appelé *cycle eulérien*. De même, un graphe possédant un tel cycle est appelé *graphe eulérien*.

Démonstration.

($\Rightarrow$) Supposons que le graphe admet un cycle eulérien. Si un sommet v apparaît k fois dans un cycle eulérien alors v est de degré $2k$ puisqu'il faut une arête incidente à v pour atteindre le sommet et une autre arête pour le quitter, ce qui prouve la parité du degré de tout sommet.

($\Leftarrow$) Soit G un graphe dont tous les sommets sont de degré pair. Prenons W = $v_0 v_1 v_2 \dots v_l$ un chemin parmi les chemins dans G de longueur maximale utilisant une arête au plus une fois, en particulier W est un chemin passant par tous les sommets de G. Puisque W ne peut être prolongé, il contient toutes les arêtes adjacentes à v_l. Par hypothèse, le nombre de telles arêtes est pair. Par conséquent, $v_l = v_0$, c'est-à-dire W est un cycle. Il reste à montrer que W passe par chaque arête. Par l'absurde, supposons que W n'est pas un cycle eulérien, c'est-à-dire qu'il existe une arête e dans G qui n'apparaît pas dans W. Puisque G est connexe, nous pouvons supposer que e est incidente à un sommet de W, disons $e = uv_i$, où u est un sommet de G. Dès lors, le chemin

$$u v_i v_{i+1} \dots v_l (= v_0) v_1 \dots v_{i-1} v_i$$

est un chemin de longueur plus grand que W, ce qui amène une contradiction. □

De manière similaire, nous pouvons nous intéresser aux cycles passant une et une seule fois par chaque sommet. De tels cycles sont appelés *cycles hamiltoniens*. De plus, un graphe *hamiltonien* est un graphe qui contient un tel cycle. À l'heure actuelle, la recherche d'un cycle hamiltonien est difficile. Dans un sens plus spécifique ce problème est NP-complet, comme nous l'informent Garey et Johnson [2, p. 199]. En effet, aucune caractérisation des cycles hamiltoniens n'a été découverte pour l'instant, contrairement aux cycles eulériens. Nous expliquons dans la section suivante la notion de complexité et plus précisément des problèmes NP-complets. Toutes ces notions se détachent du contexte de la théorie des graphes et s'inscrivent dans la théorie algorithmique.

3 Un peu (beaucoup) de complexité

Dans cette section, nous allons définir la notion de complexité ainsi que tous les outils liés à ce concept (problème algorithmique, classes de complexité, problèmes NP-complets, ...). Pour ce faire, nous nous sommes basés sur le cours de *Calculabilité et Complexité* donné par Véronique Bruyère à l'UMons, ainsi que sur les livres de Carton [3], de Garey et Johnson [2], de Rozenberg et Salomaa [4], de Teuscher [5], de Wolper [6] et une bonne introduction est également disponible sur Wikipédia [7].

3.1 Qu'est-ce que la complexité ?

La théorie de la complexité étudie et évalue la difficulté ou la complexité d'une réponse par algorithme à un problème, dit algorithmique, posé de manière mathématique. Nous allons définir trois concepts : problèmes algorithmiques, réponses algorithmiques aux problèmes et complexité des problèmes algorithmiques.

Définition 11. Un *problème algorithmique* est un problème énoncé de manière mathématique et comprenant des hypothèses, des données et une question.

Il existe deux sortes de problèmes :

- les problèmes de décision : problème où la réponse attendue est soit « oui » soit « non » ;
- les problèmes d'existence : problème qui questionne sur l'existence d'un certain élément et dont la réponse consiste à fournir un tel élément.

Par exemple, trouver un cycle eulérien dans un graphe est un problème d'existence car il faut fournir un tel cycle alors que répondre à la question « Existe-t-il un cycle eulérien ? » est un problème de décision car une réponse « oui » ou « non » est attendue. En fait, pour chaque problème d'existence, il existe un problème de décision associé.

Définition 12. Un problème possède *une réponse algorithmique* si un algorithme peut calculer la réponse à ce problème, où un algorithme est intuitivement une suite finie d'opérations permettant de résoudre un problème.

Un problème est dit *décidable* si c'est un problème de décision et si sa réponse peut être calculée par un algorithme. Par exemple pour détecter la présence d'un cycle eulérien, il suffit de vérifier si chaque sommet est de degré pair. Parallèlement, un problème est dit *calculable* si c'est un problème d'existence et si un algorithme peut donner l'élément recherché par ce problème. Pour le problème

d'existence du cycle eulérien, il existe un algorithme fournissant un tel cycle : nous partons d'un sommet x et nous effectuons un chemin ne passant qu'une seule fois par chaque arête jusqu'au moment où nous retombons sur x, ce qui est possible puisque le degré de x est pair. Si toutes les arêtes du graphe sont empruntées alors ce cycle est eulérien, sinon il existe un sommet y sur ce cycle ayant des arêtes incidentes non-empruntées. Alors, à partir de y, nous itérons le processus. Nous obtiendrons ainsi un cycle eulérien.

La théorie de la complexité se restreint aux problèmes décidables et calculables. Elle permet d'évaluer les ressources en temps et en espace mémoire utilisées pour obtenir algorithmiquement la réponse à un problème donné. La théorie de la complexité cherche à connaître si la réponse à un problème donné peut être fournie de façon très efficace ou au contraire n'est pas atteignable en pratique. Il existe évidemment différents niveaux de difficulté entre ces deux extrêmes, appelés *classes de complexité*. Pour pouvoir donner une telle information, nous nous basons sur une estimation théorique des temps de calcul et des besoins en mémoire informatique.

L'analyse de la complexité est inhérente à un modèle de calcul. L'un des modèles de calcul le plus courant est celui des machines de Turing. Une machine de Turing fonctionne selon le procédé suivant : à chaque étape, pour un état donné de la mémoire de la machine, une action élémentaire est choisie dans un ensemble d'actions possibles. Les *machines déterministes* sont celles où l'ensemble d'actions possibles est réduit à une seule tandis que pour les *machines non déterministes*, cet ensemble peut contenir plus d'une action ou aucune. Une machine de Turing permet de savoir si un mot sur un certain alphabet appartient à un *langage*, ensemble de mots particuliers définissant un problème. La machine de Turing a trois sorties différentes : un calcul acceptant, un calcul rejetant ou un calcul infini. Selon la sortie, nous pouvons peut-être déterminer si le mot appartient au langage.

Ce modèle de calcul est un précurseur des ordinateurs essayant de définir le concept d'algorithme. Comme le détaille Teuscher [5, p. 77–81], Alan Turing, mathématicien britannique du début du XX^{e} siècle, fut le premier à développer un modèle abstrait et technique permettant de calculer ce dont l'homme a les capacités. L'idée de Turing était simple : les opérations effectuées par une machine de Turing simulent, étapes par étapes, certaines opérations élémentaires. Cette découverte engendra de multiples questions, comme par exemple : « Est-ce qu'une machine peut penser ? ». Selon Turing, les machines peuvent peut-être penser mais elles ne sont pas meilleures que les humains concernant les mathématiques, ou alors elles sont peut-être meilleures que les humains mais ne savent pas penser. Beaucoup de ces questions restent encore ouvertes à l'heure actuelle. Rozenberg et Salomaa [4, p. 14–15] nous expliquent que la thèse de Church-Turing considère les machines de Turing comme formalisation de la notion d'algorithme.

Pour survoler la notion de calculabilité, nous allons parcourir les différentes classes de langages ainsi que quelques propriétés. On dit qu'un langage est *récursivement énumérable* s'il existe une machine de Turing calculant ce langage. Cette machine ne s'arrête pas nécessairement. Par contre, un langage est *récursif* s'il existe une machine de Turing qui s'arrête toujours et qui calcule ce langage. La classe des langages récursivement énumérables est notée $\mathcal{L}_{\mathrm{RE}}$ et celle des langages récursifs $\mathcal{L}_{\mathrm{R}}$. Tous les langages sont dans $\mathcal{L}$. Nous avons

$$\mathcal{L}_{\mathrm{R}} \subsetneq \mathcal{L}_{\mathrm{RE}} \subsetneq \mathcal{L}.$$

De plus, il existe certaines propositions concernant l'appartenance à une classe.

Wolper [6, p. 130–137] s'intéresse davantage à ces classes de langages et prouve ces différents résultats dans son ouvrage.

Proposition 13. *Soient* L *un langage et* L^C *son complémentaire.*

$$L \in \mathcal{L}_R \iff L^C \in \mathcal{L}_R.$$

$$L \in \mathcal{L}_R \iff L \in \mathcal{L}_{RE} \text{ et } L^C \in \mathcal{L}_{RE}.$$

$$L \in \mathcal{L}_{RE} \setminus \mathcal{L}_R \Rightarrow L^C \in \mathcal{L} \setminus \mathcal{L}_{RE}.$$

Nous allons nous intéresser uniquement à la classe des langages récursifs, $\mathcal{L}_R$, et subdiviser cette classe en plusieurs sous-classes. Auparavant, nous allons éclairer la notion de complexité. La complexité d'une machine de Turing est donnée en fonction de la taille de l'entrée et évalue le temps nécessaire à la machine de Turing pour résoudre le problème. Ce temps est calculé en nombre de transitions sur la machine de Turing, en nombre d'opérations élémentaires effectuées et ce dans le pire des cas possibles. En général, ce temps n'est pas une fonction précise mais un ordre de grandeur. Dès lors, la notion de grand O s'impose.

Définition 14. Soient f, g deux fonctions de $\mathbb{N}$ dans $\mathbb{R}$. On dit que f est *en grand* O de g, noté $f(n) = O(g(n))$ si

$$\exists C > 0, \exists N \in \mathbb{N}, \forall n > N, |f(n)| \leqslant C|g(n)|.$$

Intuitivement, cette notion signifie que f ne croît pas plus vite que g, à un facteur près. Par exemple, si $f(n) = 2n^2 + 1000n + 10^{14}$ alors f est en grand O de n^2. Si f est une fonction polynomiale de degré k alors f est en grand O de n^k. En fonction de l'ordre de grandeur, nous avons différents types de complexité :

- $O(1)$: complexité constante, indépendante de la taille des données,
- $O(\log(n))$: complexité logarithmique,
- $O(n)$: complexité linéaire,
- $O(n\log(n))$: complexité quasi-linéaire,
- $O(n^2)$: complexité quadratique,
- $O(n^3)$: complexité cubique,
- $O(n^k)$: complexité polynomiale,
- $O(2^n)$: complexité exponentielle,
- $O(n!)$: complexité factorielle.

3.2 Quelques classes de complexité

Les classes suivantes de complexité sont détaillées dans le cours de Calculabilité et Complexité donné par Véronique Bruyère. Nous n'abordons pas la complexité en espace.

- La classe TIME($t(n)$) est l'ensemble des langages récursifs L tels qu'il existe une machine de Turing déterministe calculant L et de complexité en temps $O(t(n))$.

– La classe NTIME($t(n)$) est l'ensemble des langages récursifs L tels qu'il existe une machine de Turing non déterministe calculant L et de complexité en temps O($t(n)$).
– La classe P reprend les langages récursifs pouvant être calculés par une machine de Turing déterministe en temps polynomial, c'est-à-dire

$$\mathrm{P} = \bigcup_{k \geqslant 0} \mathrm{TIME}(n^k).$$

– La classe NP est la version non déterministe de la classe P, c'est-à-dire

$$\mathrm{NP} = \bigcup_{k \geqslant 0} \mathrm{NTIME}\left(n^k\right).$$

Par exemple, le problème du cycle hamiltonien est dans la classe NP. Intuitivement, cela signifie qu'il existe un algorithme qui vérifie en temps polynomial si un chemin est un cycle hamiltonien mais jusqu'à présent, il n'existe pas d'algorithme pour trouver un tel cycle en temps polynomial. Évidemment, énumérer tous les chemins et vérifier s'ils sont cycles hamiltoniens est un algorithme trouvant un tel cycle mais sa complexité est exponentielle.
– La classe EXPTIME rassemble les langages récursifs L tels qu'il existe une machine de Turing déterministe calculant L et ayant un temps de calcul exponentiel, c'est-à-dire

$$\mathrm{EXPTIME} = \bigcup_{k \geqslant 0} \mathrm{TIME}\left(2^{n^k}\right).$$

3.3 Relations entre les différentes classes

Puisque toute machine de Turing déterministe est une machine non déterministe, il est clair que P ⊆ NP. Par le théorème suivant, déduit du livre de Wolper [6, p. 167], nous pouvons conclure que NP ⊆ EXPTIME.

Théorème 15. *Soit* M *une machine de Turing non déterministe polynomiale. On peut construire une machine de Turing déterministe acceptant le même langage que celui de* M *et ayant une complexité en* $\mathrm{O}(2^{p(n)})$ *où* $p(n)$ *est un polynôme.*

Les classes de complexité et les ensembles de langages sont disposés comme sur la figure 7. Une des grandes questions ouvertes à ce jour est : est-ce que P = NP ? En d'autres mots, pour chaque problème de la classe NP, existe-t-il un algorithme le résolvant en temps polynomial ? Nous avons l'impression que P ≠ NP mais, jusqu'à présent, aucune preuve démontrant cette inégalité n'a encore vu le jour. Par contre, nous savons que P ⊊ EXPTIME. L'argument essentiel à cette inclusion stricte réside dans le Théorème de hiérarchie que nous pouvons retrouver dans le livre de Carton [3, p. 217–218].

3.4 Théorème de Cook-Levin et problèmes NP-complets

Nous allons maintenant voir le célèbre théorème de Cook-Levin ainsi que les différents concepts qui en découlent. Garey et Johnson [2, p. 34–44] ont écrit un

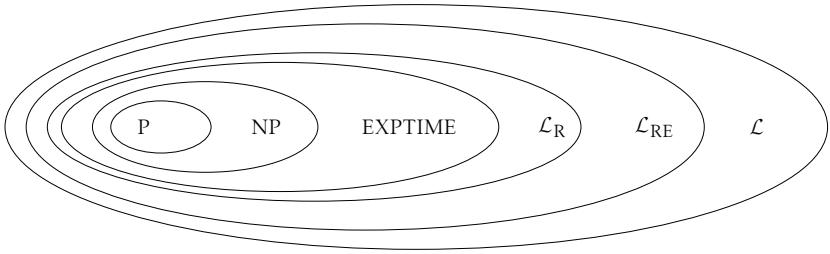


FIGURE 7 — Diagramme des différentes classes de complexité et ensembles de langages.

livre sur la complexité, considéré comme une bible de cette théorie. Ils consacrent deux sections de ce livre au théorème de Cook-Levin ce qui confirme l'importance de ce théorème. En effet, celui-ci est fondamental dans la théorie de la complexité des algorithmes. Cook l'a démontré en 1971 dans l'article [8] intitulé « The Complexity of Theorem Proving Procedures ». Nous appelons ce théorème « théorème de Cook-Levin » car il a été démontré par Levin sensiblement à la même époque. Pour comprendre la structure de la preuve de ce théorème, définissons un problème de décision particulier : le Problème SAT. Le langage associé à ce problème est l'ensemble de toutes les formules booléennes satisfaisables, c'est-à-dire des formules booléennes telles qu'il existe une assignation des variables de la formule qui la rend vraie. Par exemple, la formule booléenne, où les symboles $\vee, \wedge$ et $\neg$ représentent respectivement la disjonction (« ou »), la conjonction (« et ») et la négation,

$$\phi = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_1)$$

est satisfaisable par l'assignation $x_1 =$ vrai, $x_2 =$ faux, $x_3 =$ vrai. Par contre, aucune assignation ne rend vraie la formule suivante :

$$\phi' = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_1) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3).$$

Théorème 16 (Théorème de Cook-Levin). *Si* SAT *appartient à* P *alors* P = NP.

Pour comprendre la structure de la preuve du théorème de Cook-Levin, nous allons définir deux nouveaux concepts. Tout d'abord, certains langages récursifs ont une relation particulière entre eux : un langage peut se réduire de façon polynomiale à un autre.

Définition 17. Soient L_1, L_2 deux langages récursifs. On dit que L_1 *se réduit de façon polynomiale à* L_2, noté $L_1 \leqslant_p L_2$, s'il existe une machine de Turing déterministe C, appelée *convertisseur*, qui calcule en temps polynomial une fonction f telle que $\omega \in L_1 \iff f(\omega) \in L_2$.

Grâce à cette relation entre langages, la proposition suivante nous assure que si un langage se réduit de façon polynomiale à un autre et si cet autre est dans P, alors le premier langage est aussi dans P.

Proposition 18. *Soient* L_1, L_2 *deux langages récursifs. Si* $L_1 \leqslant_p L_2$ *et si* $L_2 \in P$ *alors* $L_1 \in P$.

Ensuite, la deuxième notion pour expliquer la structure du théorème de Cook-Levin est le langage NP-complet.

Définition 19. Un langage L est NP-*complet* si L est NP-facile, c'est-à-dire $L \in NP$, et si L est NP-difficile, c'est-à-dire $\forall L' \in NP, L' \leqslant_p L$.

Intuitivement, un langage est NP-complet s'il est dans NP et s'il est au moins aussi difficile que n'importe quel autre langage de NP. On a l'impression que les langages NP-complets sont les plus difficiles de la classe NP. La proposition suivante, qui est un élément clé dans le théorème de Cook-Levin, généralise celui-ci.

Proposition 20. *Soit* L *un langage récursif. Si* L *est* NP-*complet et si* L *est dans* P *alors* P = NP.

Cette proposition est assez importante par sa signification : il suffit de montrer qu'un seul langage NP-complet est dans P pour montrer que tous les langages de NP sont dans P. En fait, le théorème de Cook-Levin montre que SAT est NP-complet et conclut par la proposition précédente.

Montrer qu'un langage est NP-complet n'est pas une tâche facile. C'est pourquoi la proposition suivante a toute son utilité.

Proposition 21. *Soient* L_1, L_2 *deux langages récursifs. Si* L_1 *est* NP-*complet, si* L_2 *est dans* NP *et si* $L_1 \leqslant_p L_2$ *alors* L_2 *est aussi* NP-*complet*.

Jusqu'à présent, nous avons principalement parlé de langages mais, en pratique, nous parlerons essentiellement de problèmes de décision. En fait, nous avons une équivalence entre ces deux notions. Un langage forme exactement les mots sur un certain alphabet qui répond « oui » à un problème de décision. De tout problème de décision, nous pouvons extraire un langage correspondant.

Maintenant que nous avons vu les notions des problèmes P, NP et NP-complets, nous sommes dans une des deux configurations de la figure 8 mais nous ne savons pas laquelle.

3.5 Interprétation de la NP-complétude

Sous l'hypothèse que $P \neq NP$, si nous cherchons une solution algorithmique pour un problème NP-complet, alors ce problème ne possède pas de solution algorithmique polynomiale (par la contraposée de la proposition 20). Est-ce une raison pour renoncer définitivement à résoudre ce problème par un algorithme ? Les quelques arguments suivants, suggérés par Wolper [6, p. 184], nous montrent que ce n'est pas forcément le cas.

- La complexité d'un algorithme se base sur le pire des cas possibles. Dès lors, l'absence d'un algorithme polynomial se traduit par l'absence d'un algorithme dont toutes les instances du problème se calculent en un temps polynomial. Il se peut pourtant qu'il existe un algorithme dont le comportement est polynomial dans 99% des cas.
- En général, on trouve une solution à un problème NP-complet en explorant un nombre exponentiel de cas possibles. Pour limiter ce nombre, on utilise des *heuristiques*, algorithmes fournissant en temps polynomial une solution réalisable, pas nécessairement optimale. Donc, au lieu d'énumérer systématiquement toutes les possibilités, on utilise des critères approximatifs pour découvrir plus rapidement une solution recherchée. La théorie de

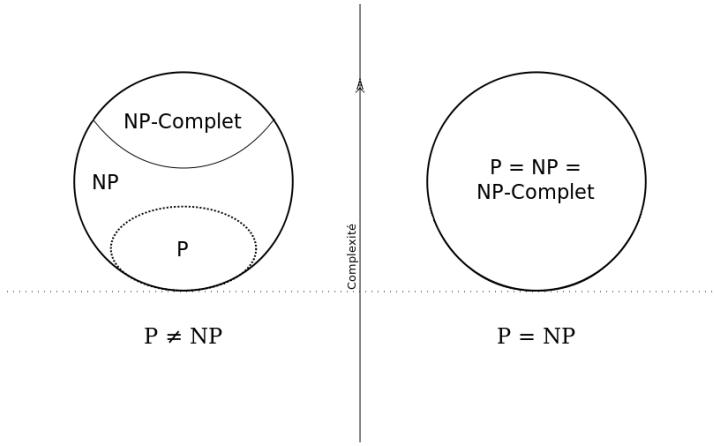


FIGURE 8 — Configurations possibles des classes de complexité.

FIGURE 9 — Provinces de Belgique.

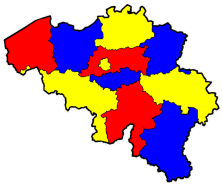


FIGURE 10 — Provinces de Belgique colorées avec 3 couleurs.

FIGURE 11 — Graphe correspondant à la carte des provinces de Belgique.

la NP-complétude nous enseigne que de telles méthodes heuristiques ne peuvent pas toujours donner de bons résultats. Cependant, rien n'exclut une certaine efficacité pour de nombreuses instances du problème.

4 Coloration de graphe

Durant les études, essentiellement pendant un cours de géographie, nous avons sûrement tous coloré une carte géographique, comme par exemple celle des provinces de la Belgique (voir figure 9). Nous sommes confrontés au problème suivant : nous devons colorer cette carte de telle manière à ce que deux provinces limitrophes aient des couleurs différentes. Étant étudiant, nous n'avons que peu de crayons de couleurs dans notre trousse. Dès lors, nous nous sommes demandés : avons-nous suffisamment de couleurs différentes pour colorer la carte ? Est-ce que nous devrons emprunter un crayon à notre voisin ? Nous pouvons, à la main, montrer qu'il faut au minimum 3 couleurs pour cette carte (voir figure 10), deux couleurs ne suffisent clairement pas.

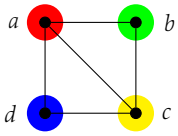
Quel est le lien entre la coloration d'une carte et un graphe ? Nous pouvons déduire le graphe suivant d'une carte : à chaque région correspond un sommet et

deux sommets sont adjacents si et seulement si leurs régions correspondantes sont limitrophes, comme illustré par la figure 11. Dès lors, colorer une carte revient à colorer un graphe selon la définition suivante.

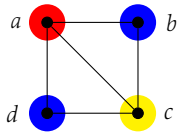
Définition 22. *Colorer* un graphe signifie assigner une couleur à chaque sommet de telle sorte que deux sommets reliés par une arête soient de couleurs différentes.

Le but dans la coloration de graphe est de trouver le nombre *minimum* de couleurs nécessaire pour colorer le graphe. Ce nombre s'appelle *nombre chromatique*. Si le graphe peut être coloré avec k couleurs, nous dirons qu'il est *k-coloriable* ou qu'il possède une *k-coloration*.

Exemple 23.



Le graphe ci-contre peut être coloré avec 4 couleurs vu qu'il possède exactement 4 sommets, mais son nombre chromatique vaut 3.



Dans la section suivante, nous allons aborder différentes applications de la coloration d'un graphe.

4.1 Applications de la coloration

Nous avons vu que la coloration d'un graphe est utile dans la coloration des cartes géographiques mais ce n'est pas là son unique domaine d'application. En effet, la coloration de graphe a de nombreuses utilités :

- les problèmes d'incompatiblité : le stockage de produits chimiques qui peuvent exploser s'ils entrent en contact, désignation d'un endroit pour des personnes ou des animaux en tenant compte des relations,
- l'allocation de fréquences, par exemple dans un réseau de téléphone mobile GSM,
- la confection d'horaires,
- la résolution du Sudoku,
- ...

Allocation de fréquences

Dans un réseau de télécommunication, il y a des émetteurs émettant chacun sur une fréquence particulière. Pour éviter les interférences sur le réseau, il ne faut pas allouer la même fréquence à deux émetteurs trop proches l'un de l'autre. De plus, à cause de contraintes financières, nous nous intéressons à minimiser le nombre de fréquences allouées. Ce problème peut être résolu par la coloration d'un graphe. En effet, il suffit de mettre un sommet pour chaque émetteur et une arête entre deux sommets si et seulement si les deux émetteurs correspondants sont trop proches, comme l'illustre la figure 12. Par conséquent, allouer des fréquences au réseau revient à colorer le graphe correspondant.

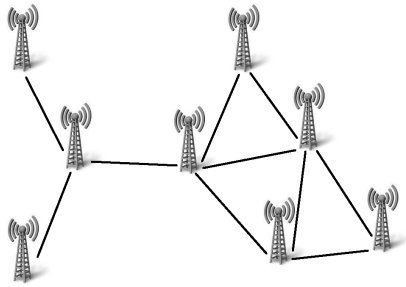


FIGURE 12 — Un réseau d'antenne sous forme de graphe.

	Math	Latin	Grec	Sciences Sociales
Classe A	X			X
Classe B	X	X		
Classe C		X	X	
Classe D		X		X

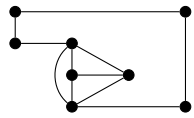


FIGURE 13 — Le problème de confection d'un horaire.

Confection d'horaires

Comment établir un horaire de telle manière que chaque classe puisse suivre son option et chaque professeur donner son cours ? Par coloration du graphe dont les sommets sont étiquetés par un cours et une classe, il y a une arête entre chaque paire de cours donnés par le même professeur ou à la même classe. Dans l'exemple de la figure 13, le professeur de Latin donne aussi le cours de Grec. Par conséquent, les sommets des cours de Latin et Grec sont tous adjacents, c'est-à-dire forment un graphe complet.

Résolution du Sudoku

Le Sudoku est un jeu dont le but est de remplir une grille 9×9 avec des chiffres différents entre 1 et 9 de sorte qu'ils n'apparaissent jamais deux fois dans une même ligne, une même colonne ou une même région. De nouveau, la résolution d'un Sudoku est exactement la coloration d'un graphe. Lequel ? À chaque cellule de la grille correspond un sommet étiqueté par un couple (x, y) où x est son numéro de ligne et y de colonne. Deux sommets (x, y) et $(\tilde{x}, \tilde{y})$ sont reliés par une arête si

9			1					5
		5		9		2		1
8				4				
				8				
			7					
				2	6			9
2			3					6
			2			9		
		1	9		4	5	7	

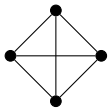
- $x = \tilde{x}$: deux cellules sont sur la même ligne,
- $y = \tilde{y}$: deux cellules sont sur la même colonne,
- $\left\lfloor \frac{x-1}{3} \right\rfloor = \left\lfloor \frac{\tilde{x}-1}{3} \right\rfloor$ et $\left\lfloor \frac{y-1}{3} \right\rfloor = \left\lfloor \frac{\tilde{y}-1}{3} \right\rfloor$: deux cellules sont dans la même région.

4.2 Graphes planaires

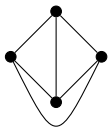
Maintenant que les motivations de la coloration sont claires, nous allons découvrir divers résultats. Parmi tous les graphes, nous allons montrer par une preuve simple que pour une certaine classe de graphes, il existe une coloration d'au plus 5 couleurs. Ce théorème important s'appelle le théorème des cinq couleurs et la classe des graphes est celle des graphes planaires.

Définition 24. Un graphe est *planaire* s'il est possible de le représenter dans le plan de sorte que les arêtes ne se coupent pas.

Exemple 25.

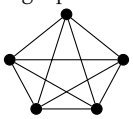


Le graphe à gauche paraît ne pas être planaire mais sa représentation à droite nous démontre le contraire. Attention à ne pas se fier à notre vue.



Toutes les cartes géographiques pour lesquelles cinq régions ne peuvent avoir une frontière commune (en un point) sont des graphes planaires.

Exemple 26. Voici deux graphes non planaires.

K_5 

$K_{3,3}$ 

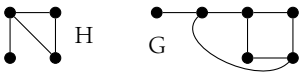
En fait, ces deux graphes sont les deux plus petits graphes non-planaires. C'est une conséquence de ce que nous dit le théorème de Kuratowski et de Wagner, comme nous l'explique Diestel [9, p. 83–101].

Théorème 27 (Théorème de Kuratowski et de Wagner). *Un graphe est planaire si et seulement s'il ne contient ni* K_5 *ni* $K_{3,3}$ *comme mineur.*

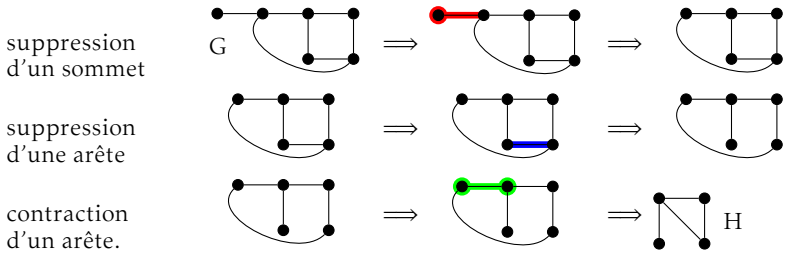
Définition 28. Un graphe H est un *mineur* d'un graphe G si H peut être obtenu en contractant les arêtes d'un sous-graphe de G, c'est-à-dire si H est obtenu en effectuant sur G les opérations suivantes dans n'importe quel ordre et en un nombre quelconque de fois :

- suppression d'une arête sans affecter ses extrémités,
- suppression d'un sommet et de toutes ses arêtes adjacentes,
- contraction d'une arête, c'est-à-dire suppression de l'arête xy, les deux sommets x et y sont fusionnés en un nouveau sommet z et toutes arêtes xu ou yu sont modifiées en zu.

Exemple 29. Le graphe H est un mineur du graphe G :



En effet, le graphe G a subi les opérations suivantes pour obtenir H :



4.3 Théorèmes de coloration des graphes planaires

Voici un premier théorème permettant de colorer tout graphe planaire. Aigner et Ziegler [10, p. 175–178] nous donnent une preuve simple de ce théorème important.

Théorème 30 (Théorème des cinq couleurs)**.** *Tout graphe planaire peut être coloré avec cinq couleurs.*

Démonstration. Pour démontrer ce théorème, remarquons tout d'abord qu'ajouter des arêtes à un graphe ne peut qu'augmenter son nombre chromatique. Par conséquent, nous pouvons supposer que $G = (V, E)$ est connexe et que toutes les faces intérieures du graphe sont bornées par un triangle, c'est-à-dire 3 arêtes forment la frontière d'une face. La figure 14 illustre cette construction.

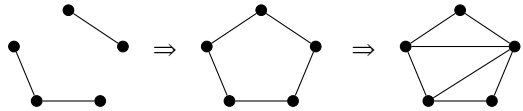


FIGURE 14 — Graphe auquel des arêtes ont été ajoutées pour obtenir un graphe connexe dont toutes les faces intérieures sont bornées par un triangle.

Posons B le cycle bordant la région extérieure et $C(v)$ une liste de couleurs potentielles de v, pour $v \in V$. Nous pouvons faire les hypothèses suivantes :

1. deux sommets adjacents fixés, disons x et y de B, sont déjà colorés avec deux couleurs différentes α et β,
2. $|C(v)| \geqslant 3$ pour tout sommet v de B différent de x et y,
3. $|C(v)| \geqslant 5$ pour tout autre sommet v, c'est-à-dire $v \in V \setminus B$.

Nous allons montrer par récurrence sur le nombre de sommets qu'il existe une 5-coloration de G à partir des listes de couleurs et en respectant les couleurs de x et de y.

Cas de base : $|V| = 3$. Ce cas est trivial puisque le seul sommet non-coloré a une liste d'au moins 3 couleurs, donc nous pouvons trouver une couleur différente de α et de β.

Étape de récurrence : $|V| > 3$. Dans un premier cas, supposons que B a une corde, c'est-à-dire une arête n'appartenant pas à B et joignant deux sommets u et v de B, voir figure 15. Posons B_1 le chemin qui est sous-graphe induit de B partant de v jusque u et passant par x et y et $B_2 = B \setminus B_1$. Le sous-graphe induit G_1 borné par $B_1 \cup \{u, v\}$ et contenant x, y, u et v est un graphe respectant les hypothèses de récurrence. Par induction, G_1 possède une 5-coloration. Supposons

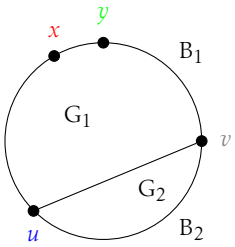


FIGURE 15 — Cas 1.

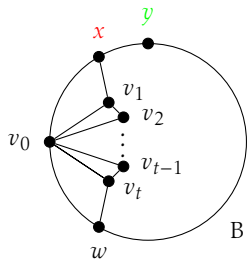


FIGURE 16 — Cas 2.

que dans cette coloration, les sommets u et v reçoivent les couleurs différentes δ et γ. Regardons G_2 le sous-graphe induit de G borné par B_2 et uv. Dans ce graphe, considérons u et v comme sommets précolorés. Par conséquent, G_2 est 5-coloriable par récurrence, et cette coloration est compatible avec la précedente. Dès lors, l'union de ces colorations colore bien G.

Dans le second cas, B n'a pas de corde. Soit v_0 le sommet de B non-coloré adjacent à x et soient $x, v_1, v_2, \ldots, v_t, w$ les voisins de v_0, où $w \in B$ et $v_i \notin B, \forall i \in \{1, 2, \ldots, t\}$.

Puisque toutes les faces internes de G sont des triangles, nous avons la situation comme illustrée dans la figure 16. Considérons $G' = G[V \setminus \{v_0\}]$, le graphe obtenu à partir de G en supprimant v_0 et toutes les arêtes adjacentes à v_0. Par conséquent, G′ a comme frontière extérieure $B' = (B \setminus \{v_0\}) \cup \{v_1, \ldots, v_t\}$. Par hypothèse, v_0 possède au moins 3 couleurs dans sa liste. Il existe donc deux couleurs dans cette liste, disons δ et γ, différentes de α. Remplaçons toutes les listes de couleurs $C(v_i)$ par $C(v_i) \setminus \{\delta, \gamma\}$. Nous pouvons vérifier que G satisfait toutes les hypothèses de récurrence et admet donc une 5-coloration. Pour finir, il suffit de choisir δ ou γ pour colorer v_0, en fonction de la couleur de w. □

En fait, il existe un théorème beaucoup plus fort que celui des cinq couleurs, démontré par Appel et Haken [11, 12] en 1976. Jusqu'ici, aucune preuve de ce théorème qui ne fait pas appel à l'ordinateur n'a été découverte. En effet, 1478 cas critiques sont étudiés via l'utilisation d'un ordinateur. Ceci explique pourquoi aucune preuve n'est présentée dans le présent article.

Théorème 31 (Théorème des 4 couleurs). *Tout graphe planaire peut être coloré avec quatre couleurs.*

Nous savons donc que tout graphe planaire admet une 4-coloration. Cependant, savoir si un graphe planaire est 3-coloriable reste un problème difficile (NP-complet). Étrangement, il est facile de connaître si le graphe peut être coloré avec 2 couleurs, autrement dit si le graphe est biparti. En effet, il existe un algorithme s'exécutant rapidement (en temps polynomial) et résolvant le problème. L'algorithme procède de la manière suivante. Tout d'abord, colorer un sommet quelconque. Ensuite, colorer les sommets voisins avec l'autre couleur et ainsi de suite. Si un sommet encore non-coloré est voisin de deux sommets de couleurs différentes alors le graphe n'est pas biparti. Cet algorithme se base sur le théorème suivant que nous démontre Diestel [9, p. 17–18]

Théorème 32. *Un graphe est biparti si et seulement si il ne contient pas de cycle de longueur impaire.*

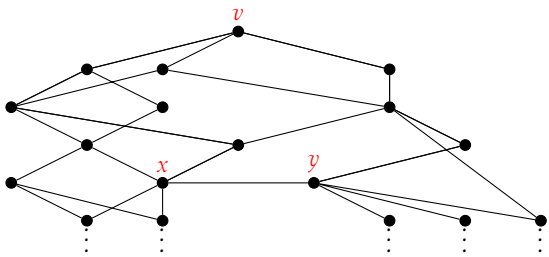


FIGURE 17 — Bipartition d'un graphe.

Démonstration.

($\Rightarrow$) Clairement, si un graphe est biparti alors il ne peut pas contenir un cycle de longueur impaire.

($\Leftarrow$) Soit G un graphe sans cycle de longueur impaire. Nous allons montrer que G est biparti, c'est-à-dire que nous pouvons le colorer avec les couleurs α et β. Soit v un sommet quelconque du graphe. Colorons-le avec α. Ensuite, colorons tous les voisins de v avec β, et les voisins des voisins de v non-colorés avec α, et ainsi de suite. Alors chaque sommet u de couleur β (respectivement α) est sur un chemin P_u de longueur impaire (respectivement paire) partant de v jusque u. Si une arête xy est dans G avec x et y de même couleur, disons α, alors yP_yvP_xxy forme un cycle de longueur impaire dans G, comme illustré dans la figure 17. Donc les couleurs α et β sont une 2-coloration de G. □

Voici une autre définition équivalente d'un graphe biparti.

Définition 33. Un graphe $G = (V, E)$ est *biparti* s'il existe une bipartition V_1, V_2 de V telle que

$$\forall xy \in E, x \in V_1 \iff y \in V_2,$$

c'est-à-dire les deux extrémités de toute arête ne sont pas toutes les deux ensemble ni dans V_1 ni dans V_2.

4.4 Conjecture de Hadwiger

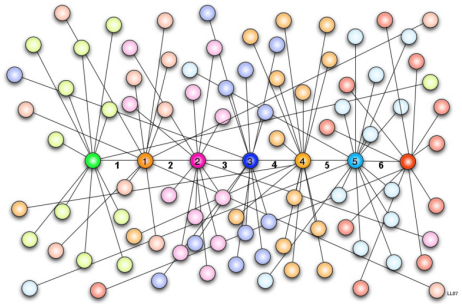
La coloration d'un graphe quelconque (non nécessairement planaire) est aussi importante que celle des graphes planaires car, nous l'avons déjà vu, la coloration de graphes a des utilités dans divers domaines. Savoir si un graphe est 2-coloriable est facile par l'agorithme (valable sur les graphes non planaires) présenté dans la section 4.3. Par contre, déterminer si un graphe est k-coloriable, avec $k > 2$ est un problème difficile (NP-complet), comme nous l'annoncent Garey et Johnson [2, p. 191]. En 1943, Hadwiger a proposé la conjecture suivante.

Conjecture de Hadwiger. Si K_n, le graphe complet à n sommets, n'est pas un mineur d'un graphe G, alors il est possible de colorer les sommets de G avec $n-1$ couleurs.

Hadwiger [13] et Dirac [14] ont démontré les cas où $n \leqslant 4$. De plus, Wagner [15] en 1937 (respectivement Robertson, Seymour et Thomas [16] en 1993) ont prouvé que le cas $n = 5$ (respectivement $n = 6$) peut se ramener au théorème des quatre couleurs. La conjecture reste ouverte pour $n > 6$.

5 Un peu de culture générale

5.1 Petit Monde et Six Degrés de Séparation

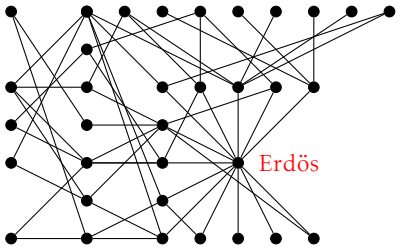


Le *phénomène du Petit Monde* est l'hypothèse que chacun est relié à n'importe quelle autre personne via une courte chaîne de relations sociales. Ce phénomène donne naissance à une théorie : les *Six Degrés de Séparation*, c'est-à-dire le fait que deux personnes peuvent être reliées par une chaîne comprenant au plus cinq autres individus. Grâce au développement des réseaux sociaux, notamment Messenger, le degré de séparation moyen parmi les personnes du réseau Messenger est mesuré précisément à 6,6 selon l'étude de Leskovec et de Horvitz [17].

5.2 Nombre d'Erdös-Bacon

Le nombre d'Erdös est un concept honorant le mathématicien Paul Erdös. Ensuite, une application de l'idée au monde du cinéma donna naissance au nombre de Bacon, en référence à l'acteur américain Kevin Bacon.

Nombre d'Erdös



La communauté des mathématiciens est fortement connectée. Le mathématicien hongrois Paul Erdös a rédigé près de 1 500 articles scientifiques durant sa vie. Il est l'un des mathématiciens ayant le plus de publications, avec Euler. Erdös a publié beaucoup d'articles alors qu'Euler a publié beaucoup de résultats. Le *nombre d'Erdös* est la distance depuis Paul Erdös en se basant sur les publications communes. Plus concrètement, Paul Erdös a son nombre d'Erdös égal à zéro, toute personne avec un nombre d'Erdös de 1 a co-écrit avec Paul Erdös, celle avec un nombre égal à 2 est co-auteur avec une personne de nombre d'Erdös de 1, etc. Les personnes n'ayant aucun lien de publication avec Paul Erdös ont un nombre d'Erdös infini.

Comme nous l'expliquent De Castro et Grossman [18], près de 500 personnes ont un nombre d'Erdös égal à 1, et plus de 5 000 ont un nombre de 2. Ce dernier nombre est sans cesse en augmentation, contrairement au nombre de personnes dont le nombre d'Erdös est égal à 1, avec la disparition de Paul Erdös en 1996.

Nombre de Bacon

Par analogie, un nombre semblable a été défini d'après l'acteur Kevin Bacon (voir figure 18) pour les acteurs jouant dans les mêmes films.

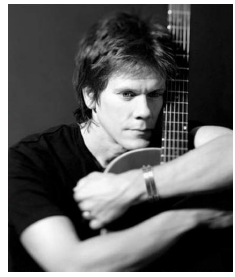
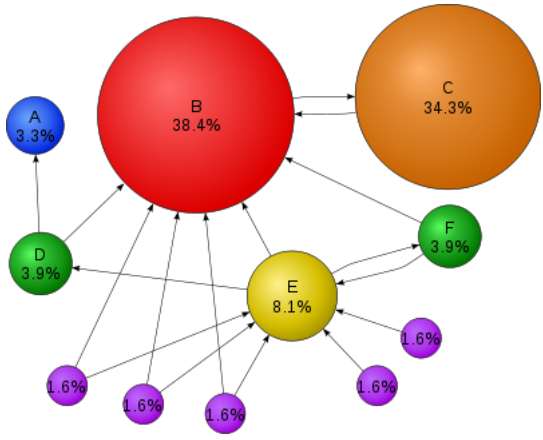


FIGURE 18 — Kevin Bacon.

Nombre d'Erdös-Bacon

Un défi auquel chacun de nous peut se soumettre est d'avoir un nombre d'Erdös-Bacon fini. Ce nombre n'est rien d'autre que la somme entre le nombre d'Erdös et celui de Bacon. En général, il est nécessaire (mais pas suffisant) d'être apparu dans un film et d'avoir co-écrit un article académique pour avoir un nombre d'Erdös-Bacon fini. L'actrice Natalie Portman possède un tel nombre fini, à savoir égal à 6, puisqu'elle a écrit un article en psychologie durant son diplôme de Harvard.

5.3 Moteur de recherche Google : PageRank



Le Web peut être représenté par un énorme graphe dans lequel chaque page est un sommet et chaque lien entre deux pages est une arête. Un des premiers principes du moteur de recherche Google est d'affecter à chaque page web une note proportionnelle au nombre de passages sur cette page d'un utilisateur explorant le graphe du Web en cliquant au hasard sur un des liens figurant sur chaque page. Cette note est appelée *PageRank*. Par conséquent, la PageRank d'une page web augmente en fonction de la somme des PageRanks des pages référant cette page. Langville et Meyer [19] expliquent comment la théorie des graphes et celle des chaînes de Markov (probabilité) permettent de gérer le moteur de recherche Google. De plus, Sarah Dendievel et Sophie Hautphenne ont fait un exposé sur ce sujet lors de la BSSM 2010 et ont publié un article sur le sujet dans [20].

6 Autres applications

La théorie des graphes est si vaste qu'un tel article ne peut parcourir tous ses résultats et toutes ses applications. Voici un début de liste d'autres domaines non-explorés.

- Trouver le plus court chemin passant par tous les sommets (problème du voyageur de commerce),
- trouver le plus court chemin entre deux sommets,

– trouver le plus long chemin entre deux sommets (gestion de projet, méthode PERT),
– trier, ordonner un ensemble d'objets selon un ordre spécifique via des algorithmes utilisant une classe particulière de graphes : les arbres,
– compresser des données sans perte grâce à certains graphes : les forêts (codage de Huffman),
– résoudre les problèmes de flot.

N'oublions pas que notre capitale belge a pour symbole une référence aux sciences. En effet, l'atomium représente le cristal de fer mais il peut aussi être vu comme un graphe !

7 Bibliographie

[1] L. Euler, « Solutio problematis ad geometriam situs pertinentis », *Commentarii academiae scientiarum Petropolitanae*, vol. 8, p. 128–140, 1741.

[2] M. Garey, D. Johnson *et al.*, *Computers and Intractability : A Guide to the Theory of NP-completeness*. wh freeman San Francisco, 1979.

[3] O. Carton, « Langages formels, calculabilité et complexité », 2008.

[4] G. Rozenberg et A. Salomaa, *Handbook of Formal Languages : Linear modeling : background and application*. Springer Verlag, 1997.

[5] C. Teuscher et D. Hofstadter, *Alan Turing : Life and legacy of a great thinker*. Springer Verlag, 2004.

[6] P. Wolper, *Introduction à la calculabilité : cours et exercices corrigés*. Sciences sup, Dunod, 2006.

[7] Wikipedia, « Théorie de la complexité des algorithmes ».

[8] S. Cook, « The complexity of theorem-proving procedures », *in Proceedings of the third annual ACM symposium on Theory of computing*, p. 158, ACM, 1971.

[9] R. Diestel, « Graph theory, Graduate Texts in Mathematics Vol. 173 », 2000.

[10] M. Aigner et G. Ziegler, « Proofs from the book », *The Australian Mathematical Society*, p. 127, 2003.

[11] K. Appel et W. Haken, « Every planar map is four colorable. part i. discharging », *Illinois J. Math*, vol. 21, no. 3, p. 429–490, 1977.

[12] K. Appel, W. Haken et J. Koch, « Every planar map is four colorable. part ii. reducibility », *Illinois J. Math.*, vol. 21, no. 3, p. 491–567, 1977.

[13] H. Hadwiger, « Uber eine klassifikation der streckenkomplexe », *Vierteljschr. Naturforsch. Ges. Zürich*, vol. 88, p. 133–142, 1943.

[14] G. Dirac, « A property of 4-chromatic graphs and some remarks on critical graphs », *Journal of the London Mathematical Society*, vol. 1, no. 1, p. 85, 1952.

[15] K. Wagner, « Über eine eigenschaft der ebenen komplexe », *Mathematische Annalen*, vol. 114, no. 1, p. 570–590, 1937.

[16] N. Robertson, P. Seymour et R. Thomas, « Hadwiger's conjecture fork 6-free graphs », *Combinatorica*, vol. 13, no. 3, p. 279–361, 1993.

[17] J. Leskovec et E. Horvitz, « Planetary-scale views on a large instant-messaging network », *in Proceeding of the 17th international conference on World Wide Web*, p. 915–924, ACM, 2008.

[18] R. De Castro et J. Grossman, « Famous trails to Paul Erdős », *The Mathematical Intelligencer*, vol. 21, no. 3, p. 51–53, 1999.

[19] A. Langville et C. Meyer, *Google page rank and beyond*. Princeton Univ Pr, 2006.

[20] S. Dendievel et S. Hautphenne, « Chaînes de Markov et Google », *in Notes de la troisième BSSM* (C. Ley, N. Richard et Y. Swan, éds), p. 15–24, 2010.