- Algorithmes paramétrés de graphes -

- École CIMPA, Mathématiques discrètes -

- Bobo-Dioulasso, octobre 2012 -

- Stéphane Bessy, Université Montpellier 2 -

Table des matières

L'objectif du cours est de décrire les techniques permettant de construire des algorithmes paramétrés 'efficaces' en algorithmique de graphes. Tous les problèmes étudiés ici auront pour entrée une instance constituée d'un graphe simple, non-orienté et possiblement d'un *paramètre*, qui sera un entier positif. On ne s'intéresse qu'aux *problèmes de décision*, c'est-à-dire aux problèmes dont la réponse est OUI ou NON.

1 Introduction

Considérons un problème Π dont l'instance est constituée d'un graphe G et possiblement d'un paramètre k. On note n le nombre de sommets de G et m son nombre d'arêtes, et on considère que $k \leq n$. La taille de l'entrée du problème Π dépend de la façon dont est codé le graphe G (liste des arêtes, liste des voisins de chaque sommet, matrice d'adjacence...) mais on considérera que la taille de l'entrée, $|G|$, vaut $n + m$.
On dit que le problème Π est *résoluble en temps polynomial* si il existe un polynôme P et un algorithme $\mathcal{A}$ qui réponde au problème avec un temps d'exécution (ou *complexité*) en $O(P(n))$ (c'est-à-dire $\leq c.P(n)$ pour une constante c). C'est le cas de nombreux problèmes classiques d'algorithmique de graphes tels que CONNEXITÉ, PLUS-COURT-CHEMIN, COUPLAGE-PARFAIT... (voir [4] pour les algorithmes usuels). On note P la classe des problèmes dont la résolution peut être fait par un algorithme polynomial.

1.1 La *NP*-complétude

La classification des problèmes suivant leur difficulté à être résolu émerge dans les années 60, notamment dans un article de J. Edmonds [7] qui fut un des premiers à se demander comment quantifier la qualité d'un algorithme.

La classe NP. Une *certification (positive)* pour un problème Π, est un algorithme $\mathcal{C}$ prenant en entrée une instance G de Π et un *certificat* C, qui est une donnée de taille polynomiale en $|G|$, et qui répond en temps polynomial en $|G|$ avec la propriété suivante : si G est une instance positive de Π, alors il existe un certificat C tel que $\mathcal{C}(G, C)$ retourne OUI. Une *certification négative* est définie de la même manière, mais la propriété à vérifier est : si G est une instance négative de Π, alors il existe un certificat C tel que $\mathcal{C}(G, C)$ retourne NON. On note *NP* (resp. *co-NP*) l'ensemble des problèmes qui admettent une certification positive (resp. négative).
La trace d'un algorithme polynomial peut être acceptée comme un certificat (positif ou négatif), ainsi, on a l'inclusion $P \subseteq NP \cap co\text{-}NP$. Deux des plus fameuses conjectures de l'informatique théorique (dues à Cook au début des années 70) concernent ces classes de problèmes et sont : est-ce que $P = NP \cap co\text{-}NP$? (peut-être que oui...) est-ce que $P = NP$? (peut-être que non...).

Exemple 1: CLIQUE, CHEMIN-HAMILTONIEN sont dans *NP*.

Réductions polynomiales et problèmes NP-complets. Un problème Π_1 se *réduit polynomialement* à un problème Π_2 si il existe une fonction f : {Instances de Π_1} $\rightarrow$ {Instances de Π_2} calculable en temps polynomial telle que : G est une instance positive de Π_1 si, seulement si, $f(G)$ est une instance positive de Π_2. La fonction f est appelée une *réduction (polynomiale)* de Π_1 à Π_2 et on note $\Pi_1 \leq_P \Pi_2$.
Un problème Π de *NP* est dit *NP-complet* si tout problème de *NP* se réduit à Π. Cela signifie que Π est un des problèmes les plus durs de *NP*.

Théorème 1 (Cook, Levin, 1971) SAT *est un problème NP-complet.*

Le théorème précédent est fondateur de la théorie de la complexité et de l'usage de celle-ci en algorithmique de graphes. Peu après, en 1972, Karp publia une liste de 21 problèmes (tels CLIQUE, CHEMIN-HAMILTONIEN, COUVERTURE, 3-COLORABLE...) qu'il montra *NP*-complets en utilisant des réduction vers SAT. La référence [13] est un 'dictionnaire' fameux de problèmes *NP*-complets, complété plus récemment par le site [16].

Exemple 2: COUVERTURE se réduit à STABLE qui se réduit à SAT et donc COUVERTURE et STABLE sont *NP*-complet.

1.2 Le théorème de Robertson et Seymour

Pour certains problèmes paramétrés comme CLIQUE ou COUVERTURE, lorsque le paramètre est fixé (on parle alors de k-CLIQUE ou de k-COUVERTURE), l'algorithme exhaustif fonctionne en temps polynomial (typiquement $O(n^k)$). Pourtant, on verra que tous les problèmes de ce type ne sont pas équivalent du point de vue de la complexité paramétrée. Une première classification a été obtenue comme un surprenant corollaire d'un théorème majeur de théorie des graphes, dû à Robertson et Seymour (voir [5] pour plus de détails).
Un graphe H est dit *mineur* de G si on peut obtenir H à partir de G en supprimant des sommets, supprimant des arêtes et/ou contractant des arêtes. Si H est mineur de G, on note $H \leq_m G$. Une classe de graphes $\mathcal{G}$ est dite *close par mineur* si pour tout graphe $G \in \mathcal{G}$ et tout mineur H de G, on a $H \in \mathcal{G}$.
Un ordre est un *bel-ordre* si tout ses sous-parties ont un nombre fini d'éléments minimaux.

Théorème 2 (Robertson, Seymour, 1985 (conjecture de Wagner, 1930)) *La classe des graphes est bel-ordonnée par la relation de mineur.*

Par ailleurs, Robertson et Seymour ont aussi montré que si H est un graphe fixé, il est possible de déterminer en temps $O(n^3)$ si H est un mineur de G ou non. Un corollaire de ces deux résultats est qu'il est possible de reconnaître une classe de graphe close par mineurs en temps polynomial.

Exemple 3: Il existe un algorithme en $O(f(k)n^3)$ pour résoudre k-COUVERTURE.

Il existe un algorithme polynomial pour résoudre GRAPHE-SANS-ENTRELAT.

1.3 La complexité paramétrée

Les applications algorithmiques du Théorème de Robertson et Seymour ont été mises en avant par Fellows et Langston à la fin des années 80 (voir [8]) et la notion d'algorithme FPT a émergé à cette époque-là. Un problème Π paramétré par un paramètre k est dit *résoluble à paramètre fixé (ou FPT pour fixed parameter tractable)* si il admet un algorithme de résolution de complexité $O(f(k).n^c)$ ou c est une constante fixée indépendante de k et f une fonction quelconque.

Exemple 4: il existe un algorithme en $O(2^k n)$ pour résoudre k-COUVERTURE.

k-COLORATION n'est pas *FPT*.

3-COLORATION paramétré par le nombre d'arêtes du graphe est *FPT*.

2 Techniques classiques pour construire des algorithmes FPT

Les techniques suivantes ont émergé au cours des années 90 et 2000 et sont décrites dans les livres classiques de complexité paramétré (voir [6], [9] et [15] ou [17] pour des documents en ligne). Par ailleurs, à la manières des recueils de problèmes *NP*-complets [13] et [16], une liste des problèmes *FPT* est dressée sur le site [10].

2.1 Arbres de branchement

La méthode la plus générale probablement pour construire un algorithme *FPT* est de construire un *algorithme de branchement*. Un tel algorithme fonctionne en deux parties : exploration de l'espace de recherche par un algorithme exhaustif (on obtient un *arbre de branchement* qui mime le comportement de l'algorithme) puis vérification pour chaque possibilité obtenue (correspondant aux feuilles de l'arbre) si une solution au problème est obtenue.
Pour réduire la complexité de l'algorithme, on cherche généralement à réduire le nombre de feuilles de l'arbre de recherche (en limitant le degré de branchement et/ou la profondeur).

Exemple 5: Il existe un algorithme en $O(1.47^k n)$ pour résoudre k-COUVERTURE.

2.2 Décompositions arborescentes

Pour résoudre la conjecture de Wagner en 1985, Robertson et Seymour ont introduit un invariant qui mesurent une forme d'arboricité d'un graphe. Pour un graphe $G = (V, E)$, une *décomposition arborescente* de G est donnée par un arbre T et une famille $(B_x)_{x \in V(T)}$ de sous-ensembles de V tels que $\bigcup_{x \in V(T)} B_x = V(G)$ et :

- pour tout arête $e = uv$ de G, il existe une partie B_x contenant u et v.
- pour tout sommet u de G, si $u \in B_x$ et $u \in B_y$, alors $u \in B_z$ pour tout sommet z sur le chemin de x à y dans T.

La *largeur* de la décomposition $(T, (B_x)_{x \in V(T)})$ est $\max\{|B_x| \ : \ x \in V(T)\} - 1$. La *largeur arborescente* (ou *tree-width*) de G, notée $tw(G)$, est la plus petite largeur d'une décomposition arborescente de G.
Calculer la largeur arborescente d'un graphe est un problème *NP*-complet. Comme la famille des graphes de largeur arborescente plus petite que k est close par mineur, il existe un algorithme *FPT* pour résoudre 'TW$\leq k$?'. Cet algorithme n'est pas implémentable, mais il existe des algorithmes plus efficaces (exponentiels (Reed 92), polynomial d'approximation à un facteur $\log n$ près (Boadlender 96), *FPT* en tw d'approximation à un facteur 4 près (Diestel 99) voir notamment [14] pour ce

dernier). En pratique, la largeur arborescente d'un graphe reste très dur à calculer.
Toutefois, si on dispose d'une décomposition arborescente de largeur t, de nombreux problèmes peuvent être traités en temps $O(f(t)n)$ grâce à de la programmation dynamique le long de la décomposition du graphe.

Exemple 6: Il existe un algorithme en $O(4^t t^4 n)$ pour résoudre STABLE-PONDÉRÉ sur les graphes de largeur arborescente inférieure ou égale à t.

Cette technique a été généralisée et formalisée par Courcelles en 1992. Une propriété de graphe est *exprimable en logique monadique du second ordre* si on peut l'écrire avec des variables pour les arêtes, les sommets, les ensembles de sommets et d'arêtes, les connecteurs logiques $\wedge$, $\vee$, $\neg$, les quantificateurs $\exists$, $\forall$ appliqués au variables, et cinq relations binaires codant l'appartenance à un ensemble (sommets ou arêtes), l'adjacence entre deux sommets, l'incidence entre un sommet et une arête et l'égalité entre variables.

Théorème 3 (Courcelles, 1992) *Tout propriété de graphe exprimable en logique monadique du second ordre peut être décider en temps FPT avec pour paramètre la largeur arborescente.*

Exemple 7: l'existence d'un cycle hamiltonien est exprimable en logique monadique du second ordre.

2.3 'Color coding'

La technique de 'color coding' a été introduite par Alon, Yuster et Zwick en 1995 (voir [1]). Cette technique est utilisée lorsqu'on cherche une sous-structure H de taille k dans un graphe. Le principe est d'améliorer l'algorithme de recherche exhaustif, qui fonctionne en $O(n^k)$, pour obtenir un algorithme *FPT* de paramètre k. Pour ça, on considère une coloration (pas forcément propre) des sommets de l'instance G en k-couleurs et on cherche une copie de H qui utilise une couleur différente pour chacun de ses sommets. Ce problème est plus simple que l'original et, très souvent, de la programmation dynamique permet d'obtenir un algorithme *FPT* pour ce problème. Ensuite, pour être sûr de trouver la copie cherchée de H, il faudrait pouvoir générer toutes les colorations des sommets de G en k couleurs. A priori, il y en a n^k et on ne gagne par rapport à l'algorithme exhaustif. Cependant, Alon, Yuster et Zwick ont montré que si on prend des colorations au hasard, on n'est pas obligé d'en considérer beaucoup, ou de façon plus déterministe, qu'on peut utiliser des colorations particulières ('k-perfect family of hash functions') introduites par Schmidt and Siegal en 1990. Une telle famille de colorations est de taille $2^{O(k)} \log^2 n$ et à la propriété que pour tout ensemble X de sommets de G de taille k, il existe une coloration c de la famille telle que tous les éléments de X ont une couleur différente par c. Ainsi, on obtient l'algorithme *FPT* voulu.

Exemple 8: le problème k-CHEMIN de paramètre k admet un algorithme *FPT*.

2.4 Compression itérative

La compression itérative est une technique popularisée par un article de Reed, Smith et Vetta de 2004 (voir [18]) dans lequel les auteurs utilisent cette technique pour déterminer si un graphe G possède un ensemble de sommets X tel que $G \setminus X$ soit un graphe biparti. Il donne un algorithme *FPT* de paramètre $|X|$ pour ce problème connu sous le nom de ODD-CYCLE-TRANSVERSAL.
Le principe de cette technique est très simple et il fonctionne pour des problèmes dans lesquels on cherche dans un graphe G si il existe un ensemble X de k sommets de G possédant une certaine propriété $\mathcal{P}$ s'exprimant sur $G \setminus X$. L'étape itérative de cette méthode, dite *compression*, est : étant donnée un ensemble X de $k+1$ sommets de G vérifiant P, peut trouver un ensemble X' de k sommets de G vérifiant P grâce à un algorithme *FPT* en k. Le corps de l'algorithme consiste ensuite à considérer un graphe G' réduit à un sommet x de G et un ensemble solution X pour G' (bien souvent $X = \{x\}$), puis d'ajouter un-à-un les autres sommets de G à G' en compressant X si besoin pour qu'il reste toujours une solution pour G'.

Exemple 9: le problème 'k-FEEDBACK-VERTEX-SET' de paramètre k admet un algorithme *FPT*.

2.5 Noyaux

L'idée principal de la construction d'un noyau est de partir d'une instance G d'un problème paramétrée Π de paramètre k et d'obtenir une instance G' équivalente à G et de taille bornée par une fonction de k. On résout ensuite le problème Π exhaustivement sur G'. Cette technique date du début de la théorie de la complexité paramétrée (début des années 1990) mais a été formalisée plus tard. Plus formellement, un *algorithme de noyau* pour un problème Π paramétré par k est une fonction calculable en temps polynomiale, qui prend en entrée un graphe G et un entier k, produit un graphe G' et un entier k' et tel qu'il existe une fonction h vérifiant :

(i) (G, k) est un instance positive de Π si et seulement (G', k') en est une

(ii) $|G'| \leq h(k)$ et $k' \leq k$

On dit alors que G' est un *noyau* de taille $h(k)$ pour Π et que Π admet un noyau de taille $h(k)$. Généralement, un algorithme à noyau est formé d'une succession de sous-algorithmes appelés des *règles de réduction*.

En fait, avoir un algorithme de noyau ou un algorithme *FPT* est équivalent.

L'étude des problèmes paramétrés admettant un noyau polynomial a connu un développement rapide ses dernières années, voir [6] pour plus de précision. De plus, les règles de réduction forment des algorithmes de pré-traitement valables d'un point de vue théorique et qui sont très efficaces en pratique.

Théorème 4 (Folklore) *Un problème paramétré admet un algorithme de résolution FPT si, et seulement si, il admet un algorithme de noyau.*

Cependant, par défaut, la taille du noyau fourni par ce théorème est exponentiel en k, le but étant d'obtenir un noyau de taille la plus petite possible. Typiquement, une question courante est de savoir si un problème paramétré par un paramètre k possède un noyau polynomial en k.

Exemple 10: k-COUVERTURE admet un noyau contenant $O(k^2)$ sommets.

3-COLORATION paramétrée par la taille d'une couverture admet un noyau de taille $t + 3^t$.

k-COUVERTURE admet un noyau contenant $O(k)$ sommets.

3 Compléments

Parmi les développements récents de la théorie des algorithmes *FPT*, on trouve les points suivant (détaillés pour la plupart dans [6]).

Choix des paramètres. Lorsqu'un problème paramétré par un invariant k admet un algorithme de résolution *FPT*, il est naturel de chercher un tel algorithme pour un paramètre k' vérifiant $k' \leq k$. Par exemple, si un algorithme admet un algorithme *FPT* pour un problème paramétré par la taille d'une couverture, en admet-il un pour le même problème paramétré par la taille d'un 'feedback-vertex-set' ?

Les paramètres les plus utilisés sont : la taille de la solution (ex : k-COUVERTURE, k-CHEMIN...), un paramètre de largeur (ex :tw), un paramètre structurel (ex : nombre d'arêtes, taille d'une couverture...), la distance à un classe de graphe simple (taille d'un ensemble X tel que $G \setminus X$ soit un stable, un biparti...) et paramètre au-delà de la garantie. Ce dernier type de paramètre correspond au cas où il est facile de trouver une borne inférieure à la taille de la solution. Le paramètre est alors la différence entre la taille de la solution et la borne inférieure connue.

Exemple 11: k-COUVERTURE-AU-DESSUS-DE-LA-GARANTIE. La question est : existe-il une couverture de G de taille $\mu + k$ où μ est la taille d'un coulage maximum de G.

Réduction paramétrée et *W*-hiérarchie. Une *réduction paramétrée* est l'équivalent d'une réduction polynomiale pour les problèmes paramétrés. Pour passer d'une instance (G_1, k_1) à une instance (G_2, k_2), on demande que le temps de la réduction soit en $f(k_1)n_1^c$ et que $k_2 = g(k_1)$ pour des fonctions f et g et une constante c. Un équivalent de la hiérarchie $P \subseteq NP$ est établi pour les

problèmes à paramètres. On définit notamment la classe $W[1]$ et les problèmes $W[1]$-difficiles. La conjecture $FPT \neq W[1]$ est l'équivalent de $P \neq NP$.

Exemple 12: k-STABLE est $W[1]$-difficile. Ainsi, k-STABLE et k-COUVERTURE ne sont pas équivalents du point de vue de la complexité paramétrée.

Non-existence de noyau polynomial. Des théorèmes ont été établis pour montrer que certains problèmes n'admettaient pas de noyau polynomial. Un *algorithme de composition* est un algorithme recevant en entrée une séquence d'instances $(G_1, k), \ldots, (G_l, k)$ pour un problème Π, s'exécutant en temps polynomial en $\sum_{i=1}^{l} |G_i| + k$ et retournant une instance (G, k') de Π telle que (G, k') est une instance positive de Π si, et seulement si, il existe i tel que (G_i, k) est une instance positive de Π et k' est polynomial en k. Sous une hypothèse de complexité raisonnable ($NP \subsetneq coNP/P$) aucun problème paramétré ayant un noyau polynomial ne possède un algorithme de composition.

Exemple 13: k-CHEMIN n'admet de noyau polynomial.
TW$\leq k$? n'admet pas de noyau polynomial.

Autres développements récents. Les *méta-heuristiques* sont des algorithmes génériques d'existence de noyau polynomial pour certains problèmes paramétrés. C'est une forme de pendant du Théorème de Robertson et Seymour ou du Théorème de Courcelles en complexité paramétrée. Voir l'article [2].
Sous une certaine hypothèse de complexité (l'*unique game conjecture*, *UGC*), il est possible de montrer que certains problèmes paramétrés n'admettent pas de noyau de taille inférieure à une certaine constante.

Exemple 14: Sous *UGC*, k-COUVERTURE n'admet de noyau de taille $(2 - \epsilon)k$ pour un certain $\epsilon > 0$.

4 Problèmes ouverts

Le site [11] contient de nombreuses actualités sur les derniers développements concernant les algorithmes *FPT*, et notamment une table [12] des meilleurs algorithmes *FPT* et noyaux de plus petite taille connus pour certains problèmes. Par exemple, on peut mentionner les problèmes ouverts suivants.

Un noyau pour ODD-CYCLE-TRANSVERSAL**.** Le problème ODD-CYCLE-TRANSVERSAL paramétré par k demande si il existe, dans un graphe G un ensemble d'au plus k sommets tels que $G \setminus X$ soit un graphe biparti. On a vu un algorithme *FPT* pour ce problème, mais aucun noyau polynomial ou preuve de la non-existence d'un tel noyau n'est connu.

Un meilleur algorithme *FPT* **pour** k-CHEMIN**.** Existe-il un algorithme *FPT* pour le problème k-CHEMIN qui fonctionne en temps $O(f(k)n^c)$ avec $f(k) = O((4 - \epsilon)^k)$ pour un certain $\epsilon > 0$?

Un noyau polynomial pour INTERVALLE-COMPLÉTION**.** Un *graphe d'intervalles* est un graphe dont les sommets sont des intervalles de la droite réelle et tel que deux sommets sont adjacents si les deux intervalles correspondants s'intersectent. Le problème INTERVALLE-COMPLÉTION de paramètre k demande si on peut ajouter au plus k arêtes à un graphe G pour en faire un graphe d'intervalles ou non. Un algorithme *FPT* est connu pour ce problème, mais l'existence d'un noyau polynomial est ouverte.

Un meilleur noyau pour 3-HITTING-SET**.** Un *hypergraphe 3-uniforme* $\mathcal{H}$ est formé d'un ensemble fini de sommets V et d'un ensemble d'hyperarêtes $\mathcal{A}$ qui sont des triplets de V. Une *couverture* de $\mathcal{H}$ est un sous ensemble de V qui intersecte toutes les arêtes de $\mathcal{H}$. Il s'agit d'une généralisation

d'une couverture d'un graphe. Le problème 3-HITTING-SET paramétré par k demande si un hypergraphe possède ou non une couverture contenant au plus k sommets. Un noyau de taille $O(k^2)$ est connu pour ce problème. La question de l'existence d'un noyau de taille $O(k^{2-\epsilon})$ pour un certain $\epsilon > 0$ est ouvertes. Une réponse à cette question utiliserait probablement des outils non encore développés aujourd'hui.

A Annexe : Quelques définitions de théorie des graphes

Pour de bons livres de référence en théorie des graphes, voir [3] ou [5].

Définitions de base. Pour un ensemble fini X, on note par $[X]^2$ l'ensemble des sous-ensembles de taille 2 de X. Un *graphe* G est une paire $(V(G), E(G))$ qui consiste en un ensemble $V(G)$, appelé *ensemble des sommets* de G, et un ensemble $E(G) \subseteq [V(G)]^2$, appelé l'*ensemble des arêtes* de G. Classiquement, on note $n = |V(G)|$ et $m = |E(G)|$. Pour simplifier les notations, on note uv la paire non-ordonnée $\{u, v\}$ de $E(G)$. Deux sommets distincts x et y qui appartiennent à une même arête e sont dits *adjacents*, x et y sont les *extrémités* de e et x et y sont adjacents.
L'ensemble des sommets qui sont adjacents à un sommet x est appelé le *voisinage* de x et est noté $N_G(x)$. Ainsi, les sommets de $N_G(x)$ sont appelés les *voisins* de x. Le *degré* d'un sommet x, noté $d_G(x)$, est le cardinal de $N_G(x)$. La *matrice d'adjacence* d'un graphe G est la matrice A_G définie sur $V(G) \times V(G)$ par $A_{(x,y)} = 1$ si $xy \in E(G)$ et 0 sinon.
Un graphe $H = (V(H), E(H))$ est un sous-graphe de $G = (V(G), E(G))$ si $V(H) \subseteq V(G)$ et $E(H) \subseteq V(G)$. Si H est un sous-graphe de G avec $V(H) = V(G)$, on dit que H est un sous-graphe *couvrant* de G. Et si H est un sous-graphe de G avec $E(H) = E(G) \cap [V(H)]^2$, on dit que H est un sous-graphe *induit* de G. Pour X un sous-ensemble de $V(G)$, le *graphe induit par G sur X*, noté par $G[X]$, est le graphe induit de G qui a X comme ensemble de sommets. On note aussi par $G \setminus X$ le sous-graphe induit par G sur $V(G) \setminus X$. Et, si F est un sous-ensemble de $E(G)$, on note par $G - F$ le sous-graphe de G d'ensemble de sommets $V(G)$ et d'ensemble d'arêtes $E(G) \setminus F$. Finalement, un graphe G est isomorphe à un graphe H si il existe une bijection $f : V(G) \to V(H)$ telle que $xy \in E(G)$ si, et seulement si, $f(x)f(y) \in E(H)$.

Quelques graphes particulier. Le *graphe complet* sur n sommets, noté par K_n est le graphe d'ensemble de sommets $V(K_n) = \{v_1, \ldots, v_n\}$, et d'ensemble d'arêtes $[V(K_n)]^2$. Une *clique* d'un graphe G est un sous-ensemble de $V(G)$ qui induit un graphe complet sur G. De façon similaire, le *graphe vide* sur n sommets (qui n'est pas totalement vide...) est le graphe sur n sommets sans arête. Un *stable* ou *ensemble indépendant* est un sous-ensemble de $V(G)$ qui induit un graphe vide sur G. Le *graphe chemin* sur k sommets, noté par P_k, est le graphe d'ensemble de sommets $V(P_k) = \{v_1, \ldots, v_k\}$, et d'ensemble d'arêtes $\{v_i v_{i+1} \ : \ i = 1, \ldots, k-1\}$. Les sommets v_1 et v_k sont appelés les *extrémités* de P_k. Un *chemin* d'un graphe G est un sous-graphe de G qui est isomorphe à un graphe chemin. Un *graphe cycle* sur k sommets, noté par C_k est le graphe sur les sommets $\{v_1, \ldots, v_k\}$ dont les arêtes sont $\{v_i v_{i+1} \ : \ i = 1, \ldots, k-1\} \cup \{v_1 v_k\}$. Un *cycle* d'un graphe G est un sous-graphe de G qui est isomorphe à un graphe cycle. Parfois, on parle de *k-cycle* pour préciser que le cycle considéré a k sommets. Un graphe *hamiltonien* est un graphe qui a un cycle couvrant. Un graphe *acyclique* est un graphe sans cycle. Pour finir, un *couplage* est un graphe dont les arêtes sont deux-à-deux disjointes.

Maintenant, à l'aide de ces graphes particuliers, on peut définir quelques propriétés sur les graphes. Un graphe est *connexe* si pour toute paire, x et y, de ses sommets il existe un chemin d'extrémités x et y. Un *arbre* est un graphe connexe sans cycle. Il est bien connu qu'un graphe est connexe si, et seulement si, il contient un arbre couvrant. Un graphe est *biparti* si son ensemble de sommets peut se partitionner en deux stables. Enfin, le graphe *biparti complet*, $K_{p,q}$, est le graphe de sommets $\{v_1, \ldots, v_p, w_1, \ldots, w_q\}$ et d'arêtes $\{v_i w_j \ : \ i = 1, \ldots, p \text{ and } j = 1, \ldots, q\}$.

Quelques invariants de graphes. La *stabilité* d'un graphe G, notée $\alpha(G)$, est la taille d'un

plus grand stable de G. De même, le *nombre de clique* d'un graphe G, notée $w(G)$, est la taille d'une plus grande clique de G. Une *k-coloration (propre)* d'un graphe G est une fonction c de $V(G)$ vers l'ensemble $\{1, \ldots, k\}$ tel que, si xy est une arête de G, alors $c(x) \neq c(y)$. De manière équivalente, une k-coloration de G est une partition de son ensemble de sommets en k stables. Le *nombre chromatique* de G, noté $\chi(G)$ est le nombre k minimum pour lequel G admet une k-coloration. Les graphes bipartis sont exactement les graphes G vérifiant $\chi(G) \leq 2$. Par ailleurs, les sommets d'une clique de G doivent tous avoir une couleur différentes dans une coloration de G et on a donc $w(G) \leq \chi(G)$.

Références

[1] N. Alon, R. Yuster, U. Zwick, Color-coding, *J. ACM* **42-4** : 844–856, 1995.

[2] H.L. Bodlaender, F.V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D.M. Thilikos, (Meta) Kernelization, *Proc. of 50th FOCS* : 629–638, 2009.

[3] J.A. Bondy, U.S.R. Murty, Graph Theory, *Springer*, 2008.

[4] T. Cormen, C. Leiserson, R. Rivest, Introduction à l'algorithmique, *Dunod Ed.*, 1994.

[5] R. Diestel, Graph Theory, *Springer*, 2010.

[6] R. Downey, M. Fellows, Parameterized Complexity, *Springer*, 1999

[7] J. Edmonds, Paths, trees, and flowers. *Canad. J. Math.* **17** : 449–467, 1965.

[8] M.R. Fellows, M.A. Langston, Nonconstructive Advances in Polynomial-Time Complexity, *Inf. Process. Lett. (IPL)* **26(3)** :155-162 (1987).

[9] J. Flum, M. Grohe, Parameterized complexity theory, *Springer*, 2006.

[10] Compendium de problèmes *FPT* : `http://www.compendium-fpt.com/`

[11] Site d'actualités sur les algorithmes *FPT* :`http://fpt.wikidot.com/welcome`

[12] Liste des meilleurs algorithmes *FPT* pour certains problèmes : `http://fpt.wikidot.com/fpt-races`

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

[14] J. Kleinberg, E. Tardos, Algorithm Design. *Addison Wesley*, 2005.

[15] R. Niedermeier, Invitation to fixed-parameter algorithms, *Oxford University Press*, 2006.

[16] Compendium de problèmes *NP*-complet : `http://www.nada.kth.se/~viggo/problemlist/compendium.html`

[17] Page de C. Paul : `http://www.lirmm.fr/~paul/`

[18] B. Reed, K. Smith, A. Vetta, Finding odd cycle transversals, *Operation Research Letters* **32** : 299–301, 2004.