Recherche combinatoire de graphes

Mathilde Bouvel

Stage effectué au Loria, équipe ADAGE,
sous la direction de Gregory Kucherov

Mars - Août 2005

La recherche combinatoire s'intéresse à l'identification d'un objet inconnu ou « caché » dans un ensemble d'objets, à l'aide de questions indirectes sur cet objet. Cette branche de l'algorithmique combinatoire est étroitement liée à d'autres domaines informatiques et mathématiques comme l'étude de la complexité des algorithmes, la théorie des nombres ou la théorie des graphes. Les terrains applicatifs des problèmes de recherche combinatoire sont nombreux et variés. Ceux qui vont nous intéresser sont principalement la médecine et la biologie.

La recherche combinatoire sur les graphes présente de nombreuses facettes. Nous verrons dans la suite plusieurs problèmes génériques classiques, et nous nous concentrerons sur le problème de recherche d'un graphe caché dans une classe de graphes donnée. Un point commun à tous les types de problèmes envisagés en recherche combinatoire sur les graphes est le suivant : les questions posées portent sur des ensembles de sommets, et les réponses obtenues fournissent des informations sur les arêtes du graphe caché.

Ce rapport de stage est organisé comme suit. Après une brève description du contexte dans lequel j'ai effectué mon stage, une partie introductive présente les motivations historiques de la recherche combinatoire, et quelques définitions et outils fréquemment utilisés. Ensuite, un état de l'art expose des résultats connus de recherche combinatoire sans se limiter aux graphes. Nombre de ces résultats seront utilisés dans la dernière partie qui concerne à proprement parler la recherche combinatoire de graphes.

Table des matières

1 Contexte du stage

Mon stage de deuxième année s'est effectué au Loria, unité mixte CNRS et INRIA à Nancy, du mois de mars au mois d'août 2005. J'étais encadrée par Gregory Kucherov, directeur de l'équipe ADAGE (Algorithmique Discrète Appliquée à la GEnomique). Dans cette équipe, les principaux axes de recherche sont l'algorithmique du texte et la géométrie discrète, avec la bioinformatique comme domaine d'application privilégié, mais non exclusif.

La recherche combinatoire a été un thème de recherche actif dans l'équipe [8, 9, 10, 11, 12], notamment entre 1995 et 1998, période au cours de laquelle Vladimir Grebinski a préparé, sous la direction de Gregory Kucherov, son doctorat intitulé « Recherche Combinatoire : Problèmes de Pesage, Reconstruction de Graphes et Applications »[9]. Dans ce travail, les problèmes de recherche combinatoire étudiés sont motivés par des problèmes biologiques réels, et la majeure partie de cette thèse est consacrée à la recherche combinatoire sur les graphes.

Gregory Kucherov était conférencier invité au WG 2005, et il lui avait été demandé de préparer un exposé portant à la fois sur la théorie des graphes et sur la bioinformatique. C'est donc tout naturellement qu'il a voulu réutiliser le travail fait avec son ancien étudiant. Mon rôle pendant le stage a été de l'assister dans la préparation de cette conférence, de l'aider à se replonger dans ses anciens travaux, de trouver dans la littérature des articles récents traitant du même thème et de construire un ensemble cohérent à partir de tous ces éléments. J'ai pû en outre assister aux conférences qui ont eu lieu à Metz du 23 au 25 juin 2005 et participer à la rédaction d'un article qui paraîtra dans les actes de ce congrès.

2 Introduction à la recherche combinatoire

Citons ici deux ouvrages de référence qui pourront être consultés par ceux souhaitant plus d'information sur les sujets évoqués : *Combinatorial Search* de M. Aigner [1], où les problèmes de recherche combinatoire de graphes sont aussi traités ; et *Combinatorial Group Testing and its Applications* de D.-Z. Du et F.K. Hwang [7].

2.1 Un peu d'histoire

L'ouvrage [7] présente en introduction un bref historique du *group testing* et de la recherche combinatoire. Voici en quelques mots le récit de la naissance de cette discipline.

Les pesages de monnaie Les premiers problèmes de recherche combinatoire qui sont apparus concernent les problèmes de pesage de pièces (*group testing* en anglais). Ils ont été étudiés par les mathématiciens au début du

vingtième siècle, mais ont probablement été un soucis majeur des banquiers de tout temps. L'idée de départ de ces problèmes est de retrouver, dans un ensemble de pièces de monnaie, une ou plusieurs fausse(s) pièce(s). Pour cela, on dispose d'une balance, et on suppose que le poids d'une fausse pièce diffère de celui d'une pièce authentique, qui est par ailleurs connu. À chaque pesée, on évalue le poids d'un ensemble de pièces et on peut ainsi savoir si l'ensemble contient au moins une fausse pièce, ou bien connaître le nombre de fausses pièces dans l'ensemble pesé dans certains modèles. Le but des mathématiciens, mais sans doute des banquiers avant eux, est de retrouver la ou les fausse(s) pièce(s) en un nombre de pesées le plus petit possible.

Les tests d'échantillons de sang Les problèmes de *group testing* ne sont pas un sujet très en vogue dans la première moitié du vingtième siècle. Ils réapparaîssent dans les années 1942–1943 pour résoudre un problème auquel était confrontée l'armée américaine au cours de la Seconde Guerre Mondiale. Afin de détecter seulement quelques milliers de cas de syphilis parmi tous les soldats, l'armée américaine a décidé de prélever des échantillons de sang sur ses soldats pour les soumettre à des tests permettant de trouver les échantillons contaminés. Un groupe d'économistes et de statisticiens de Washington, parmi lesquels Robert Dorfman et David Rosenblatt, est très surpris par le gaspillage d'autant de tests pour détecter une petite proportion de malades. Le test effectué étant sensible à une quantité même très faible de sang contaminé, l'un d'eux propose de mélanger les échantillons pour économiser des tests. C'est ainsi qu'apparaît sous sa première forme l'idée du group testing. On teste des « super-échantillons » de sang, mélanges de plusieurs échantillons prélevés dans la population de soldats. Si un test révèle la présence de l'antigène de la maladie dans un « super-échantillon » de sang, il faut retester chaque échantillon d'origine, pour identifier le ou les soldats malades. Un test est alors gaspillé. Mais dans le cas où l'antigène n'est pas détecté, beaucoup de tests sont économisés. Tout le problème est de savoir si cette stratégie d'organisation des tests est plus intéressante que la stratégie naïve. On peut aussi refléchir à d'autres stratégies pour organiser les tests qui sont plus efficaces encore, ou chercher quel serait le nombre optimal de tests. Robert Dorfman a mené une étude statistique et publié une note à ce sujet dans Annals of Mathematical Statistics. C'est ainsi que la communauté scientifique a commencé à s'intéresser aux problèmes de *group testing*.

La recherche combinatoire À ses débuts, le *group testing* n'était envisagé que d'un point de vue probabiliste et statistique, qui s'intéresse donc à l'étude des phénomènes *en moyenne*. Peu à peu, une étude plus combinatoire s'est parallèlement imposée, le *combinatorial group testing*, qui fait quant à lui la part belle à *l'étude au pire des cas* et à la *conception d'algorithmes*

efficaces pour résoudre les problèmes concrets. C'est cette deuxième orientation qui, s'ouvrant à des problèmes plus larges que les problèmes de pesage de monnaie[1], est devenue ce que l'on appelle aujourd'hui communément la *recherche combinatoire.*

2.2 Formalisation du problème

Un problème de recherche combinatoire peut être décrit par la donnée de deux ensembles finis $\mathcal{C}$ et $\mathcal{D}$ et d'un sous-ensemble $\mathcal{Q}$ des fonctions de $\mathcal{C}$ dans $\mathcal{D}$. $\mathcal{C}$ est appelé l'ensemble des *configurations.* $\mathcal{Q}$ est l'ensemble des *questions* et $\mathcal{D}$ l'ensemble des *réponses possibles*, qu'on pourra toujours identifier à $\{0, \ldots, d-1\}$ pour un certain $d \geq 2$. Il est essentiel de remarquer ici que la *nature des questions* est une donnée intégrante du problème ; en changeant le type des questions que l'on s'autorise à poser, on ne traite plus tout à fait le même problème.

Exemple 1 *Sur l'exemple des pesages de pièces, les configurations sont des lots de pièces dont certaines sont authentiques et certaines sont fausses, les questions sont des pesées de tout ou partie des pièces de la configuration, et on peut imaginer que les réponses sont* `oui` *ou* `non` *selon que la pesée révèle la présence ou l'absence de fausse(s) pièce(s). On pourrait aussi imaginer que la balance ne puisse supporter que des pesées d'au plus 10 pièces à la fois, ce qui restreindrait l'ensemble des questions et constituerait donc un autre problème.*

Exemple du nombre caché 1 *Un jeu célèbre qui donne un problème simple de recherche combinatoire est le jeu du nombre caché. Les configurations sont les entiers x entre* 1 *et* 1000*. Les réponses possibles sont* `oui` *et* `non`*. Et les questions sont de la forme « x est-il inférieur ou égal à y ? », pour tous les y entre* 1 *et* 999*.*

Dans beaucoup de cas, les configurations seront simplement des « objets cachés » que l'on cherche à retrouver, comme dans le jeu du nombre caché.

Le but de la recherche combinatoire est de discriminer les configurations entre elles, de pouvoir distinguer les objets cachés les uns des autres. Dans l'exemple précédent, cela revient à trouver quel était le nombre caché. Pour formaliser cette idée, on introduit quelques définitions.

Définition 1 *Pour une configuration $x^* \in \mathcal{C}$, on appelle* séquence de recherche *toute suite finie de la forme $(q_1(x^*), q_2(x^*), \ldots, q_k(x^*))$ où $q_i \in \mathcal{Q}$ pour tout i, $1 \leq i \leq k$. On dit qu'une séquence de recherche est* déterminante *lorsque le k-uplet $(q_1(x), q_2(x), \ldots, q_k(x))$ est différent de $(q_1(x^*), q_2(x^*), \ldots, q_k(x^*))$ pour tout $x \in \mathcal{C} \setminus \{x^*\}$.*

[1] Retrouver des personnes malades dans une population n'est pas autre chose que retrouver des fausses pièces dans un ensemble de pièces.

Il est important de remarquer que dans une séquence, le choix de chaque question peut *a priori* dépendre des réponses précédentes. Nous y reviendrons dans la sous-section 2.4.

Définition 2 *La* solution *d'un problème de recherche combinatoire est un algorithme qui, pour toute configuration x, choisit la question q_{k+1} étant donnée $(q_1(x), q_2(x), \ldots, q_k(x))$, et tel que pour toute configuration x, il existe un entier n tel que la séquence $(q_1(x), q_2(x), \ldots, q_n(x))$ soit déterminante.*

Pour étudier ce type d'algorithme, on utilise une méthode particulière d'évaluation de la complexité. Étant donné un algorithme f, pour une configuration x, notons $c_f(x)$ le plus petit entier n tel qu'il existe une séquence déterminante $(q_1(x), q_2(x), \ldots, q_n(x))$ calculée par f pour x. La complexité de l'algorithme f est le maximum des $c_f(x)$ pour $x \in \mathcal{C}$.

Définition 3 *La complexité d'un problème de recherche combinatoire est la complexité minimale d'un algorithme résolvant ce problème.*

La complexité d'un problème désigne, dans le cadre de la recherche combinatoire, le *nombre de questions* (ou *requêtes*) qu'il faut effectuer dans le pire des cas pour discriminer une configuration des autres. On ne mesure pas, dans l'évaluation de la complexité, les calculs que l'on doit faire à partir des réponses aux questions pour retrouver la configuration cachée. Ceci peut être justifié par le fait qu'il est toujours possible, au moins en théorie, de précalculer une table qui fait correspondre à un ensemble de réponses obtenues la configuration associée.

Une autre manière d'expliquer cette notion de complexité est d'imaginer un jeu entre deux joueurs A et B. A choisit une configuration et B pose des questions à A pour essayer de deviner laquelle. La complexité du problème est le nombre de questions *nécessaires* à B pour découvrir la configuration choisie par A, dans le pire des cas. En supposant que B joue selon une stratégie optimale, la complexité est le nombre maximal de questions que B pose, le maximum étant pris sur l'ensemble des parties possibles, c'est-à-dire sur l'ensemble des choix possibles de A.

Remarquons qu'il est en général difficile de calculer exactement la complexité d'un problème et on se contente le plus souvent d'encadrements. La sous-section 2.5 présente des méthodes classiques pour calculer des bornes inférieures et supérieures sur les complexités.

2.3 Modèles booléen et quantitatif

Les différents types de questions que l'on peut poser sont regroupés dans deux grandes familles : il s'agit des modèles *booléen* et *quantitatif* (ou *additif*) de questions.

Dans le modèle booléen, l'ensemble des réponses possibles contient seulement deux éléments : `oui` et `non`, 0 et 1, ... L'information obtenue est

booléenne, ce qui confère son nom à ce premier modèle générique de questions.

Exemple du nombre caché 2 *Étendons le jeu du nombre caché au jeu des* 10 *nombres cachés : on doit découvrir* 10 *nombres entre* 1 *et* 1000*. Pour un y fixé, la question qui sera posée est « Y a-t-il au moins un des nombres cachés qui est inférieur ou égal à y ? ».*

Dans le modèle quantitatif en revanche, chaque réponse fournit une information *quantitative* sur la configuration étudiée, le plus souvent un nombre de « choses » contenues dans la requête, la chose en question dépendant du problème considéré. C'est un modèle *a priori* plus fort que le modèle booléen.

Exemple du nombre caché 3 *Dans le jeu des* 10 *nombres cachés, le modèle quantitatif autorise des questions du type, pour un y fixé, «* Combien *y a-t-il de nombres cachés qui sont inférieurs ou égaux à y ? ».*

2.4 Algorithmes adaptatifs et non-adaptatifs

On distingue principalement deux types d'algorithmes dans le cadre de la recherche combinatoire.

Les algorithmes adaptatifs : Pour un algorithme de ce type, toute requête peut dépendre des réponses obtenues aux requêtes précédentes et de l'information que l'on a pu en tirer quant à la configuration étudiée. En quelque sorte, l'algorithme « s'adapte » aux réponses déjà connues.

Exemple du nombre caché 4 *Une simple recherche dichotomique est un algorithme adaptatif qui permet, en au plus* 10 *questions, de trouver le nombre caché* $x \in \{1, \ldots, 1000\}$.

Il est souvent important d'éviter la dépendance inhérente à ce type d'algorithmes, ce qui nous conduit à en considérer un autre genre.

Les algorithmes non-adaptatifs : Dans ce cas, les requêtes doivent être indépendantes les unes des autres, et peuvent par conséquent être menées en parallèle, ce qui a un interêt en pratique.

Il est aussi intéressant de considérer parfois des algorithmes adaptatifs d'un type particulier, qui sont composés d'une succession d'étapes, chaque étape étant non-adaptative. Lorsque s étapes non-adaptatives se succèdent, on parle d'algorithmes à s tours (*s-round algorithms*).

Les algorithmes non-adaptatifs sont un cas particulier d'algorithmes à s tours, et ces algorithmes sont eux-même un cas particulier d'algorithmes

adaptatifs.

On peut aussi comparer les complexités d'algorithmes adaptatifs et non-adaptatifs résolvant le même problème. Dans certains cas, et contre toute attente, les meilleurs algorithmes connus sont non-adaptatifs, et on sait même parfois qu'ils sont de complexité optimale à un facteur multiplicatif près. Dans ce genre de situation, l'adaptativité n'apporte pas ou très peu d'expressivité. Dans d'autres cas en revanche, on peut démontrer que les algorithmes non-adaptatifs sont strictement moins puissants que les algorithmes adaptatifs.

2.5 Outils génériques pour l'évaluation de la complexité

Une manière simple d'obtenir des bornes inférieures sur la complexité d'un problème est d'utiliser une astuce issue de la théorie de l'information. Cela permet de trouver une borne inférieure de manière systématique pour chaque problème, cette borne n'étant malheureusement souvent pas optimale.

Le calcul de cette borne repose sur l'idée suivante. Supposons que nous ayons n configurations et d réponses possibles aux questions. En imaginant que l'on numérote en binaire les configurations, chacune d'entre elles est caractérisée par $\log_2 n$ bits d'information. Chaque réponse à une question permet de découvrir au mieux $\log_2 d$ bits d'information. Le nombre de questions nécessaires est donc borné inférieurement par $\frac{\log_2 n}{\log_2 d} = \log_d n$.

Remarquons que dans le calcul de cette borne inférieure, on ne tient absolument pas compte du type de questions considéré dans le problème. Celà explique en partie pourquoi les bornes obtenues peuvent être très loin des bornes inférieures optimales.

Exemple du nombre caché 5 *Un algorithme* non-adaptatif *qui résoud le problème du nombre caché avec des questions du type « x est-il inférieur ou égal à y ? » ne peut pas avoir une complexité inférieure à* 999. *En revanche, si les questions autorisées sont « que vaut le i-ième bit de x ? »,* 10 *questions suffisent.*

La source la plus importante de bornes supérieures est la construction d'un algorithme explicite de résolution. On s'attendrait à ce que les algorithmes fournissant les meilleures bornes supérieures de complexité soient adaptatifs, mais cela n'est pas toujours le cas. Pour certains problèmes, les meilleurs algorithmes connus sont non-adaptatifs, ou à s tours.

Une autre manière de calculer des bornes supérieures ou inférieures sur la complexité des problèmes de recherche combinatoire est de faire appel à la méthode probabiliste ou à la méthode d'estimation de la variance. Nous

ne les utiliserons pas dans la suite mais, faisant référence à des résultats démontrés grâce à ces méthodes, nous les mentionnons pour mémoire, avec une référence standard associée [4].

3 État de l'art

Les problèmes de pesage de pièces constituent la première partie de cet état de l'art. Les résultats qui y sont présentés ont une importance considérable dans l'établissement de résultats pour la reconstruction de certaines classes de graphes, notamment pour les circuits Hamiltoniens. Nous y reviendrons en section 4.3 et 4.4.

La deuxième partie présente un aperçu des différents types de problèmes que l'on peut rencontrer en recherche combinatoire sur les graphes. Dans la section 4, nous nous concentrerons sur un type de problème particulier.

3.1 Problèmes de pesage de pièces

Il existe plusieurs problèmes de pesage de pièces, selon les modèles de questions autorisées, et aussi selon les informations que l'on cherche à retrouver sur les pièces de monnaie. Mais dans tous les cas, chaque question porte sur un *sous-ensemble* de l'ensemble des pièces donné en entrée. Ceci fait qu'un algorithme non-adaptatif n'est en fait qu'un ensemble de sous-ensembles de l'ensemble des pièces. Il peut donc être représenté par une *matrice d'incidence objet-question*. Dans cette matrice M, chaque colonne représente une pièce, et chaque ligne une question. Le coefficient $M_{i,j}$ vaut 1 si l'objet j appartient à la question i, et vaut 0 sinon. La complexité d'un algorithme non-adaptatif est le nombre de lignes de sa matrice d'incidence objet-question. Cette représentation matricielle s'avère très utile pour formaliser les problèmes de pesage de pièces étudiés, et pour étudier leur complexité.

Dans tout ce qui suit, l'ensemble des pièces donné en entrée sera assimilé à $V = \{1, \ldots, n\}$, chaque pièce étant authentique ou fausse. Pour une question qui est un sous-ensemble Q de V, la réponse obtenue est l'appartenance ou non d'au moins une fausse pièce à Q si on travaille dans le modèle booléen, ou le nombre de fausses pièces appartenant à Q si on est dans le modèle quantitatif. Nous considérerons essentiellement le modèle quantitatif dans cette partie.

3.1.1 Détermination de d fausses pièces et matrices de d-séparation

Examinons d'abord le cas où l'on recherche une fausse pièce dans l'ensemble de pièces $V = \{1, \ldots, n\}$. Dans ce cas, le modèle quantitatif n'apporte pas d'information supplémentaire par rapport au modèle booléen. En effet, lorsque l'on questionne un sous-ensemble $Q \subseteq V$, la réponse obtenue est

`vrai` ou `faux` dans le modèle booléen, 1 ou 0 dans le modèle quantitatif, selon que la fausse pièce appartient ou non à Q.

Chaque configuration correspond à un choix possible d'une fausse pièce ; il y en a donc n. Comme il y a deux réponses possibles, la borne inférieure fournie par la théorie de l'information pour la complexité de ce problème est $\lceil \log_2 n \rceil$. Comme dans le jeu du nombre caché, une recherche par dichotomie permet de trouver la fausse pièce en $\lceil \log_2 n \rceil$ questions. La recherche dichotomique a cependant le défaut d'être une méthode complètement adaptative. On peut alors se demander s'il existe un algorithme non-adaptatif de même complexité. Même si cela semble moins évident, la réponse est oui, et l'algorithme non-adaptatif sous-jacent est simple : on considère la représentation en binaire des nombres de 1 à n qui représentent les pièces, et on pose les questions $Q_i = \{j |$ le i-ième bit de j est égal à $1\}$ pour i allant de 1 à $\lceil \log_2 n \rceil$. Ainsi, la réponse à chaque question Q_i donne la valeur du i-ième bit de la fausse pièce.

Considérons maintenant le problème général de retrouver d fausses pièces parmi n, dans le modèle quantitatif. Pour chaque question $Q \subseteq V$, la réponse est le nombre de fausses pièces contenues dans Q. Pour formaliser les configurations qu'il nous faut discriminer, on peut utiliser le modèle suivant : une configuration est un vecteur de $\{0,1\}^n$ contenant exactement d 1. Un 1 à la i-ième position indique que la pièce numéro i est fausse, un 0 qu'elle est authentique. Il y a $\binom{n}{d}$ configurations, et $d+1$ réponses possibles (les nombres de 0 à d). La borne inférieure de la théorie de l'information pour la complexité de ce problème est donc $\log_{d+1} \binom{n}{d} = (1+o(1))\frac{d}{\log d} \log n$. Il est possible en utilisant la méthode d'estimation de la variance d'améliorer cette borne inférieure en $(2+o(1))\frac{d}{\log d} \log n$.

Un algorithme adaptatif simple constitué de d étapes de recherche dichotomique permet de retrouver les d fausses pièces en $d \log_2 n$ questions, ce qui n'atteint pas la borne inférieure de la théorie de l'information. Avec l'utilisation de codes correcteurs d'erreurs, il est aussi possible d'atteindre une complexité de $d \log_2 n$ questions pour un algorithme *non-adaptatif* qui retrouve d fausses pièces parmi n. Une autre construction explicite d'un algorithme non-adaptatif avec $d \log_2 n$ questions est proposée dans [9]. Elle fait intervenir des outils algébriques sur les corps finis.

Le problème de savoir si $d \log_2 n$ était la complexité optimale d'un algorithme résolvant le problème de recherche de d fausses pièces parmi n a été ouvert pendant une trentaine d'année. Bernt Linström, dont nous reparlerons plus tard, conjecture en 1975 [15] que cette borne inférieure est optimale. Vladimir Grebinski infirme cette conjecture dans sa thèse.

Théorème 1 *Il existe un algorithme de complexité* $\mathcal{O}(\frac{d}{\log d} \log n)$ *qui résoud le problème de recherche de d fausses pièces parmi n. Cette complexité atteint la borne inférieure fournie par la théorie de l'information et est donc*

optimale à un facteur multiplicatif près.

On peut en outre remarquer que cet algorithme est *non-adaptatif*. Le théorème ci-dessus est en fait un cas particulier d'un résultat plus général sur la reconstruction de vecteurs à poids borné, dont la preuve est *non constructive*. Il utilise la représentation sous forme matricielle d'algorithmes non-adaptatifs. Une matrice d'incidence objet-question qui résoud le problème de recherche de d fausses pièces parmi n s'appelle une matrice de d-séparation. Ces matrices possèdent des propriétés intéressantes qui rendent leur étude plus aisée que l'analyse directe des algorithmes sous-jacents.

Il est important de remarquer que ce résultat d'existence ne permet pas de *construire* une matrice de d-séparation ayant $\mathcal{O}(\frac{d}{\log d}\log n)$ lignes. Ce problème est encore *ouvert* à l'heure actuelle. C'est même l'un des principaux problèmes ouverts concernant les pesages de pièces.

3.1.2 Matrices de d-détection

Le problème que l'on considère ici est une extension du problème de recherche d'un nombre *arbitaire* de fausses pièces parmi n.

Dans la recherche d'un nombre arbitraire de fausses pièces, une modélisation des configurations est, comme dans le problème précédent, de considérer qu'une configuration est un vecteur de $\{0,1\}^n$, la présence d'un 1 à la i-ième position indiquant que la i-ième pièce est fausse.

L'extension consiste à imaginer que les pièces ne sont plus seulement authentiques ou fausses, mais qu'elles ont différents *degrés de fausseté*. On considèrera de manière générale d degrés de fausseté, allant de 0 pour les pièces authentiques à $d-1$ pour les pièces « les plus fausses ». Une configuration est donc un vecteur de $\{0,\ldots,d-1\}^n$ où la i-ième coordonnée du vecteur indique le degré de fausseté de la i-ième pièce. Le problème initial de recherche d'un nombre arbitraire de fausses pièces est simplement le cas particulier $d=2$.

Pour la reconstruction de tels vecteurs dans le modèle quantitatif, les questions sont du type suivant : pour un sous-ensemble Q de pièces, la réponse obtenue est la somme des degrés de fausseté des pièces contenues dans le sous-ensemble Q. Dans ce problème, l'approche à l'aide de matrices d'incidence objet-question est encore très adaptée. Une matrice qui permet de reconstruire un vecteur de degrés de fausseté compris entre 0 et $d-1$ est appelée une matrice de d-détection. Le théorème suivant est prouvé par Bernt Lindström dans [14, 15] pour le cas particulier $d=2$ et la preuve est étendue au cas général dans [12].

Théorème 2 *Le nombre de lignes d'une matrice de d-détection est borné inférieurement par $(2+o(1))\log d\frac{n}{\log n}$, et une matrice de d-détection avec $(2+o(1))\log d\frac{n}{\log n}$ lignes peut être effectivement construite.*

La preuve de la borne inférieure est probabiliste et fait appel à une méthode d'estimation de la variance d'une variable aléatoire. La construction effective de Lindström est ingénieuse et utilise des propriétés de la fonction de Möbius. Cette construction s'étend au cas d quelconque.

Sans donner les détails de la construction, donnons sur un exemple l'idée de Lindström dans le cas $d = 2$.

Exemple 2 *Supposons qu'on veuille retrouver les fausses pièces qui sont en nombre arbitraire parmi 4 pièces. Les valeurs de x_1, x_2, x_3 et x_4 indiquent si les pièces dont les numéros sont en indice sont authentiques (valeur 0) ou fausses (valeur 1). La matrice M ci-dessous est la matrice d'incidence objet-question d'un algorithme non-adaptatif permettant de déterminer les fausses pièces.*

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

En effet,

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \end{pmatrix} \quad \begin{array}{l|l} e_1 + e_2 - e_3 = 2x_3 + x_4 & x_3, x_4 \\ e_1 = x_1 + x_3 + x_4 & x_1 \\ e_2 = x_2 + x_3 & x_2 \end{array}$$

Ainsi, la valeur de $2x_3 + x_4$, *qui peut être obtenue en combinant les trois réponses aux questions,* détermine x_3 et x_4 de manière unique, *puisque $x_3, x_4 \in \{0, 1\}$. Les deux premières questions permettent ensuite de retrouver les valeurs de x_1 et x_2.*

3.1.3 Reconstruction de vecteurs à poids borné

Précisons d'abord que nous considérons ici seulement des vecteurs à coordonnées entières positives et que nous employons le terme *poids d'un vecteur* pour désigner la somme de ses coordonnées.

Le problème de reconstruction de vecteurs à poids borné est une généralisation des deux problèmes de pesage de pièces présentés avant. En effet, la recherche de d fausses pièces parmi n est un cas particulier de la reconstruction d'un vecteur de longueur n et de poids d; et dans la recherche des degrés de fausseté compris entre 0 et $d-1$ de n pièces, chaque coordonnée du vecteur est bornée par $d-1$, et leur somme est donc bornée par $n(d-1)$. Pour généraliser, on considère l'ensemble de tous les vecteurs de dimension n et de poids borné par d, noté $\Lambda(n, d)$:

$$\Lambda(n, d) = \{(v_1, \ldots, v_n) | \quad v_i \in \mathbb{N} \quad \text{et} \quad \sum_{i=1}^{n} v_i \leq d\}$$

Le problème qu'on se pose est d'estimer la complexité d'un algorithme *non-adaptatif* qui résoudrait le problème de recherche combinatoire dont les caractéristiques sont les suivantes. Les configurations à discriminer sont les vecteurs $(v_1, \ldots, v_n)$ de $\Lambda(n, d)$. Les questions portent sur des sous-ensembles de coordonnées $Q \subseteq \{1, \ldots, n\}$. Et la réponse à une question Q est la somme des valeurs associées aux coordonnées de Q : $\sum_{i \in Q} v_i$. Les vecteurs que l'on considère étant de poids borné par d, les réponses possibles (les entiers de 0 à d) sont bien en nombre fini.

Remarquons qu'un algorithme non-adaptatif naïf résoud ce problème en n questions : $Q_i = \{i\}$ pour $1 \leq i \leq n$.

Dans le théorème ci-dessous, les bornes supérieures et inférieures démontrées dans [9] ne sont pas obtenues de manière constructive. La borne supérieure est établie par la méthode probabiliste, et la borne inférieure par la méthode d'estimation de la variance.

Théorème 3 *La complexité de la reconstruction de vecteurs de longueur n et de poids borné par d est comprise entre* $(4 + o(1))\frac{min(n,d) \cdot \log \frac{max(n,d)}{min(n,d)}}{\log min(n,d)}$ *et* $(2 + o(1))\frac{min(n,d) \cdot \log \frac{max(n,d)}{min(n,d)}}{\log min(n,d)}$.

Dans [9], on trouve aussi une méthode effective de construction d'une matrice représentant un algorithme non-adaptatif reconstruisant un vecteur à poids borné. Cette méthode utilise la décomposition d'un nombre en produit de facteurs premiers. Le nombre de lignes de la matrice construite, c'est-à-dire la complexité de l'algorithme associé *au sens de la recherche combinatoire* n'est pas optimale : cet algorithme fait $d \log_2 n$ requêtes.

3.2 La recherche combinatoire de graphes

Précisons dès à présent que les graphes que l'on considère sont toujours des *graphes non-orientés* et dont les sommets sont *étiquetés* par les entiers allant de 1 au nombre n de sommets du graphe. On admet aussi qu'il n'y a pas d'arêtes reliant un sommet à lui-même.

Dans la recherche combinatoire sur les graphes, de nombreux problèmes peuvent être envisagés, et il existe aussi de nombreux modèles qui peuvent servir de cadre à l'étude de ces problèmes. Un point commun à la très grande majorité des modèles et des problèmes est que les questions portent sur des sous-ensembles de *sommets* mais permettent d'obtenir des informations concernant les *arcs*.

Les problèmes de recherche combinatoire sur les graphes peuvent être vus comme une généralisation des problèmes de pesage de pièces. Les pièces seraient les *arcs*, mais les questions ne peuvent porter que sur un certain type de sous-ensembles d'arcs : les sous-graphes complets induits par des sous-ensembles de sommets.

On présente brièvement dans la suite les principales caratéristiques et les résultats essentiels concernant trois types de problèmes de recherche combinatoire sur les graphes. Un panorama plus large peut être trouvé dans [1, 7].

3.2.1 Recherche d'un arc dans un graphe donné

Dans ce premier problème, un graphe $G = (V, E)$ est donné en entrée. Il est fixé, *connu à l'avance*. Dans ce graphe, on recherche un arc e^* qui est « défectueux ». Les configurations correspondent donc aux différents arcs défectueux e^* possibles. Les questions portent sur des sous-ensembles de sommets $Q \subseteq V$. La réponse retournée est `vrai` si $e^* \in Q \times Q$, `faux` sinon. La différence essentielle avec les problèmes de pesage de pièces est que l'on ne peut pas tester n'importe quel sous-ensemble d'arêtes, mais seulement ceux de la forme $Q \times Q$ pour un sous-ensemble Q de sommets. Il est à noter enfin que le nombre de sommets dans le sous-ensemble Q sur lequel porte la requête n'est pas limité *a priori*.

Dans [5], les auteurs démontrent que le nombre minimal de requêtes nécessaires pour résoudre ce problème est borné inférieurement par $\lceil \log_2 |E| \rceil$ et supérieurement par $\lceil \log_2 |E| \rceil + 3$. Un résultat plus fort apparaît dans [6] : la borne supérieure est réduite à $\lceil \log_2 |E| \rceil + 1$. En outre, pour une infinité de graphes, on sait que $\lceil \log_2 |E| \rceil + 1$ questions sont nécessaires. Dans cet article, la preuve permet de construire un algorithme simple et efficace qui, sur l'entrée G, produit une stratégie pour mener les tests afin de retrouver l'arête défectueuse, la stratégie produite faisant au plus $\lceil \log_2 |E| \rceil + 1$ tests. Pour chaque graphe G, on est donc en mesure de calculer un algorithme (au sens de la recherche combinatoire, c'est-à-dire une série de tests) qui retrouve une arête défectueuse en un nombre de tests optimal à un test près.

3.2.2 Reconstruction de graphes dans une classe donnée

Dans ce problème, une classe de graphes $\mathcal{G}$ est fixée et est partitionnée en $\mathcal{G} = \cup_n \mathcal{G}_n$, chaque $\mathcal{G}_n$ contenant exactement tous les graphes de $\mathcal{G}$ sur l'ensemble de sommets $V = \{1, \ldots, n\}$. Pour un entier n fixé, on cherche à reconstruire entièrement un graphe G de $\mathcal{G}_n$ que l'on appelle le graphe caché. On veut en fait découvrir quel est le graphe G de $\mathcal{G}_n$ qui a été fixé par un oracle, un autre joueur, un mécanisme biologique, ou autre chose, mais qui nous est inconnu. Les questions autorisées sont très restreintes : elles sont du type « l'arc (i, j) appartient-il à G ? » pour deux sommets i et j de V.

On considèrera dans la suite des variantes de ce problème, où l'ensemble des questions autorisées est plus vaste. Elles peuvent porter sur des sous-ensembles de sommets d'une taille quelconque, pas seulement sur deux sommets. Dans le modèle booléen, les questions autorisées seront du type « Pour $Q \subseteq V$, y a-t-il au moins une arête du graphe caché reliant deux

sommets de Q ? ». Dans le modèle quantitatif, on autorisera les requêtes « Pour $Q \subseteq V$, combien y a-t-il d'arête(s) du graphe caché reliant deux sommets de Q ? ». *Ce sont ces deux variantes du problème initial que l'on étudiera dans la dernière section de ce travail.*

La complexité de ce type de problème dépend de la structure combinatoire de $\mathcal{G}_n$. Elle est une fonction de n, mais peut aussi dépendre dans certains cas d'autres paramètres [2].

Dans le premier modèle de questions, où on ne peut que tester des paires de sommets pour savoir si une arête les relie, la complexité des problèmes de reconstruction de graphes est quadratique pour la plupart des classes de graphes que l'on examine habituellement [1] comme les couplages, les arbres, les graphes bipartis, les chemins Hamiltoniens, les circuits Hamiltoniens.

3.2.3 Vérification de propriétés de graphes

Comme dans le problème précédent, une classe $\mathcal{G} = \cup_n \mathcal{G}_n$ de graphes est fixée et connue. Un graphe G est aussi fixé, et il nous est comme avant inconnu. Mais dans ce problème, on ne cherche pas à reconstruire entièrement G en sachant que $G \in \mathcal{G}_n$. On veut seulement décider si G appartient à $\mathcal{G}_n$ ou non, sans nécessairement reconstruire G complètement. Les questions permettent dans ce problème encore de tester si une paire de sommets induit une arête ou non : pour $i, j \in V$, on peut demander « l'arc (i, j) appartient-il à G ? ».

La classe $\mathcal{G}$ est supposée représenter une propriété de graphes, c'est-à-dire contenir exactement tous les graphes vérifiant une certaine propriété. Formellement, une propriété de graphe est une fonction booléenne à $\binom{n}{2}$ arguments $f(x_{i,j})_{1 \leq i < j \leq n}$ telle que pour toute permutation $\pi \in S_n$, on a $f(x_{\{1,2\}}, \ldots, x_{\{i,j\}} \ldots) = f(x_{\{\pi(1),\pi(2)\}}, \ldots, x_{\{\pi(i),\pi(j)\}} \ldots)$. Si l'on note $f(G)$ la fonction booléenne $f(x_{i,j})$ avec $x_{i,j} = 1$ ssi $\{i, j\}$ est une arête de G, la condition précédente explique seulement que la propriété de graphe doit être indépendante de la numérotation des sommets.

4 Reconstruction de graphes dans une classe donnée

Les résultats présentés dans cette partie détaillent le deuxième problème de recherche de combinatoire de graphes présenté plus haut, pour certaines classes de graphes particulières. D'abord, on présentera les résultats concernant la reconstruction graphes à degré borné et de circuits Hamiltoniens obtenus par Vladimir Grebinski dans sa thèse. Ensuite, on considèrera les classes des couplages, des étoiles et des cliques, dont l'étude a été menée pour certains modèles seulement dans des articles récents [2, 3]. Dans cette partie apparaissent les résultats obtenus pendant le stage, qui envisagent les problèmes déjà posés dans [2, 3] sous d'autres modèles.

4.1 Motivations biologiques

Considérons le problème biologique générique suivant : supposons que nous disposons d'un ensemble de produits chimiques (des molécules par exemple) et que certaines paires de produits chimiques peuvent provoquer une réaction si on les mélange. Supposons aussi que l'on dispose d'un mécanisme expérimental qui permette de détecter si une réaction se produit dans un mélange de produits chimiques, ou même de compter le nombre de telles réactions. Le but des biologistes est de retrouver toutes les paires de produits chimiques qui réagissent, en faisant le moins d'expériences possible, celles-ci ayant toujours un coût à ne pas négliger.

Un problème biologique réel de ce type apparaît dans les projets de séquençage complet d'un génome circulaire, où des fragments contigus de ce génome qu'on a réussi à séquencer dans une première étape (les *contigs*) sont séparés par des trous, et ont un placement les uns par rapport aux autres sur le génome qui est inconnu. Les biologistes ont besoin, pour achever le séquençage, de trouver ce placement relatif des contigs entre eux, c'est-à-dire l'ordre et l'orientation des contigs sur le cercle du génome circulaire. Chaque extrémité d'un contig est caractérisée par un fragment d'ADN que l'on appelle *primer*. Les primers et les contigs sont des candidats possibles pour jouer le rôle des produits chimiques évoqués dans la formulation générique du problème. Des réactions chimiques appelées PCR (*Polymerase Chain Reaction*) se produisent entre les primers qui sont situés de part et d'autre d'un trou (et aussi entre les contigs qui sont le support de tels primers). Lorsque le nombre de contigs est petit (inférieur à dix), il est possible en pratique de mener les expériences biologiques pour tester toutes les paires de primers afin de déterminer lesquelles réagissent entre elles. Cependant, il arrive qu'il y ait plus de contigs, et dans ce cas les expériences seraient trop nombreuses si on testait toutes les paires de primers. Au lieu de cela, on fait donc appel à la technique du *multiplex PCR*, qui permet de tester plus de deux primers à la fois, et de voir parmi les primers testés si des réactions se produisent, ou même de compter combien de réactions se produisent. Il faut donc, pour rendre cette méthode efficace, trouver un protocole expérimental qui permette de retrouver les paires de primers qui réagissent avec le moins d'expériences possibles. C'est typiquement un problème de recherche combinatoire !

Le problème générique expliqué au début se modélise naturellement comme un problème de recherche combinatoire sur les graphes, plus précisement un problème de recherche de graphe dans une classe de graphes donnée. Les produits chimiques sont représentés par les sommets, et les réactions entre deux produits chimiques par une arête les reliant. Retrouver toutes les paires de produits chimiques qui provoquent une réaction est exactement la même chose que de reconstruire entièrement le graphe associé.

Pour le problème particulier lié au séquençage du génome, deux modèles sont possibles, selon que l'on considère comme produits chimiques les contigs ou les primers.

Considérons d'abord que les sommets du graphe à reconstruire représentent les contigs [10]. Une fois trouvé l'ordre des contigs sur le génome, on peut trouver leur orientation grâce à un nombre linéaire de réactions PCR. Dans ce premier modèle, on s'attache donc seulement à trouver l'"ordre circulaire" qui représente le placement des contigs sur le cercle. Un tel ordre n'est pas autre chose qu'un *circuit Hamiltonien*[2], avec des sommets représentant les contigs et les arcs les réactions PCR entre contigs.

D'un autre point de vue [3], on peut imaginer que les produits chimiques avec lesquels on travaille sont les primers. Dans ce cas, il suffit de trouver, pour chaque primer, le primer qui lui fait fasse de l'autre côté du trou dans le génome en partie séquencé. La connaissance de cette correspondance entre les primers permet de retrouver directement l'ordre et l'orientation des contigs sur le génome circulaire. Dans ce modèle, le problème sous-jacent est la reconstruction d'un graphe pris dans la classe des *couplages*[2], ou plus précisement des *couplages parfaits*[2].

4.2 Définitions

On rappelle dans cette partie quelques définitions de théorie des graphes, qui décrivent des classes de graphes dont on va étudier la reconstruction par la suite. Les premières définitions sont très classiques, mais les deux dernières, qui sont plus spécifiques des problèmes étudiés, méritent un peu d'attention.

Un *graphe* G est défini par un couple d'ensembles $G = (V, E)$, avec $E \subseteq \binom{V}{2}$, où $\binom{V}{2}$ désigne l'ensemble des paires d'éléments de V, les paires étant non-ordonnées. On assimilera en général V à $\{1, \ldots, n\}$ pour $n = |V|$. Nous supposerons toujours dans ce travail que pour tout $x \in V$, $(x, x) \notin E$. V est appelé l'ensemble des *sommets* et E l'ensemble des *arêtes* ou *arcs*. Pour tout $v \in V$, on appelle *voisins de* v ou *adjacents à* v les sommets $w \in V$ tels que $\{v, w\} \in E$. On appelle enfin *degré* de $v \in V$ le nombre de voisins de v.

Définition 4 (Graphe r-parti) *Un graphe $G = (V, E)$ est dit* r-parti *si V admet une partition en r classes, telle que toute arête ait ses deux sommets dans deux classes différentes : les sommets qui sont dans la même classe ne sont pas adjacents.*

Dans le cas où $r = 2$, on parle de graphe *biparti* plutôt que 2-parti.

[2]Les définitions de ces termes de théorie des graphes sont rappelées dans la sous-section suivante.

Une *étoile* est un cas particulier de graphe biparti : une classe contient un unique sommet x appelé *centre*, et l'autre classe contient tous les autres sommets, appelés *feuilles*, qui sont tous adjacents à x.

Définition 5 (Cycle Hamiltonien) *Un* cycle Hamiltonien *est un graphe G dont les arêtes forment un cycle passant exactement une fois par chaque sommet de G.*

Définition 6 (Couplage) *Un* couplage *est un graphe dans lequel le degré de tout sommet est* 0 *ou* 1*.*

Définition 7 (Couplage parfait) *Un* couplage parfait *est un graphe dans lequel le degré de tout sommet est exactement* 1 *si le nombre de sommets est pair. Lorsque ce nombre est impair, c'est un graphe dans lequel un unique sommet a pour degré* 0 *et les autres ont pour degré* 1*.*

Définition 8 (Graphe complet) *Un graphe $G = (V, E)$ est* complet *lorsque tous les sommets de V sont deux à deux adjacents : $E = \binom{V}{2}$. On note en général K_n le graphe complet à n sommets.*

On trouve aussi le terme de *clique* pour désigner un graphe complet.

Définition 9 (Sous-graphe engendré) *Dans un graphe $G = (V, E)$, le* sous-graphe engendré *par un sous-ensemble de sommets $S \subseteq V$ est le graphe $G_S = (S, E \cap (S \times S))$.*

Définition 10 (Graphe k-dégénéré) *Un graphe $G = (V, E)$ est* k-dégénéré *s'il existe un ordre sur les sommets $V = (v_1, \ldots, v_n)$ tel que le degré de v_i est inférieur ou égal à k dans le sous-graphe de G engendré par les sommets $\{v_i, v_{i+1}, \ldots, v_n\}$.*

De manière plus intuitive, un graphe $G = (V, E)$ est k-dégénéré s'il existe un sommet de degré inférieur ou égal à k dans G et si $G_{V \setminus \{v\}}$ possède la même propriété. Le graphe $G_{V \setminus \{v\}}$ est identique à G, sauf qu'on lui a retiré le sommet v ainsi que toutes les arêtes contenant v.

Remarquons qu'il existe de nombreuses sous-classes intéressantes des graphes k-dégénérés : citons les arbres, qui sont 1-dégénérés, les graphes planaires ou les graphes à degré borné.

Définition 11 (Classes *Stars* et *Cliques*) *Pour un entier n fixé, on considère les ensembles Stars et Cliques de graphes sur n sommets.*
Notons S_k l'ensemble des graphes constitués d'une étoile à k feuilles et de $n-k-1$ sommets isolés, et C_k l'ensemble des graphes composés d'une clique à k sommets et de $n-k$ sommets isolés.
On définit alors $Stars = \cup_{k=0}^{n-1} S_k$ et $Cliques = \cup_{k=1}^{n} C_k$.

4.3 Graphes à degré borné et graphes k-dégénérés dans le modèle quantitatif

Un lien important entre les problèmes de pesage de pièces et la reconstruction de graphes est mis en évidence dans la thèse de Vladimir Grebinski [9]. Le résultat principal, qu'il obtient en utilisant les résultats sur les vecteurs à poids borné présentés avant, concerne la reconstruction de graphes k-dégénérés. Le théorème 6 est en fait une généralisation d'un résultat sur la reconstruction de graphes à degré borné (théorème 5). Dans la suite, nous verrons les grandes idées qui tissent les preuves de ces deux résultats.

Rappelons d'abord le type de questions que l'on est autorisé à poser dans la reconstruction de graphes dans le modèle quantitatif. Pour un certain sous-ensemble Q de sommets, on peut demander le nombre d'arêtes du graphe caché G qui relient deux sommets de Q. On notera la réponse $\mu_G(Q) = |E \cap (Q \times Q)|$. On utilisera la *représentation bipartie* des graphes.

Définition 12 *Étant donné un graphe $G = (V, E)$, la* représentation bipartie *de $G = (V, E)$ est $G' = (V_1, V_2; E')$, où V_1 et V_2 sont deux copies disjointes de V et où $\{i, j\} \in E \Leftrightarrow (i, j) \in E'$ et $(j, i) \in E'$.*

Lorsque l'on travaille avec un graphe biparti $G' = (V_1, V_2; E')$, on autorise des questions du type $\mu'_{G'}(X, Y) = |E' \cap (X \times Y)|$. Il est important de remarquer que les questions de ce type portant sur un graphe G' qui est la représentation bipartie de G peuvent être simulées par un nombre constant de questions du type $\mu_G(Q)$ sur G. On a en effet, par de simples considérations ensemblistes, l'identité suivante :
$\mu'_{G'}(X, Y) = \mu_G((X \backslash Y) \cup (Y \backslash X)) - 2\mu_G(X \backslash Y) - 2\mu_G(Y \backslash X) + \mu_G(X) + \mu_G(Y)$

À partir de cette remarque, on peut aisément énoncer le théorème suivant :

Théorème 4 *Considérons une classe de graphes $\mathcal{G}$ et la classe $\mathcal{G}'$ des representations biparties des graphes de $\mathcal{G}$. Si on dispose d'un algorithme de reconstruction de graphes pour la classe $\mathcal{G}'$, alors on peut en déduire un algorithme de reconstruction de graphes pour la classe $\mathcal{G}$, qui a la même complexité à un facteur multiplicatif près.*

Cette remarque nous permet de nous limiter à l'étude des *représentations biparties* des graphes à degré borné ou k-dégénérés plutôt que ces graphes en eux-même.

On considère le problème de reconstruire un graphe biparti à degré borné dans une des deux parties au moins. Donnons-nous un graphe $G = (V_1, V_2; E')$ tel que $deg(v) \leq d$ pour tout $v \in V_1$. Supposons en outre que $|V_1| = |V_2| = n$, hypothèses naturelles puisque les graphes qui nous intéressent sont des représentations biparties de graphes à degré borné. Fixons un $v \in V_1$. Les questions $\mu'_G(\{v\}, Q)$ pour $Q \subseteq V_2$ indiquent le

nombre de voisins de v qui se trouvent dans Q. Si on traite les voisins de v comme des fausses pièces et les autres sommets de V_2 comme des pièces authentiques, on peut donc simuler une recherche de au plus d fausses pièces parmi n pour retrouver tous les voisins de v. Ce problème peut être résolu en $\mathcal{O}(\frac{d}{\log d}\log n)$ requêtes non-adaptatives (théorème 1). Il serait alors immédiat de reconstruire le graphe $G = (V_1, V_2; E')$ en $\mathcal{O}(nd\frac{\log n}{\log d})$ requêtes, puisque la donnée des voisins de v pour tout $v \in V_1$ détermine le graphe G. Mais il est possible d'améliorer cette complexité de reconstruction ! L'idée est en fait de ne pas chercher les voisins de v pour chaque $v \in V_1$ indépendamment, mais de le faire pour tous les v en même temps.

De manière plus formelle, le théorème 1 nous fournit un ensemble de m questions non-adaptatives sous la forme de sous-ensembles de $V_2 : P_1, \ldots, P_m$, qui permettent de reconstruire un vecteur de 0 et de 1 de longueur n avec au plus d 1. On sait aussi par ce théorème que $m = \mathcal{O}(\frac{d}{\log d}\log n)$. Il suffit, pour reconstruire le graphe G, de déterminer les valeurs des $\mu'_G(\{v\}, P_j)$ pour tous les $v \in V_1$ et tous les j entre 1 et m. Fixons un j entre 1 et m. En utilisant des matrices de d-detection (théorème 2), on peut reconstruire le vecteur $(\mu'_G(\{v_1\}, P_j), \ldots, \mu'_G(\{v_n\}, P_j))$, où les coordonnées sont inférieures à d, grâce à ℓ requêtes non-adaptatives $\mu'_G(S, P_j)$ pour ℓ sous-ensembles S de V_1 que l'on notera $S_1, \ldots, S_\ell \subseteq V_1$. Ce théorème 2 nous donne aussi l'estimation $\ell = \mathcal{O}(\log d\frac{n}{\log n})$. Ainsi, on peut reconstruire tout le graphe G en posant les requêtes $\mu'_G(S_k, P_j)$ pour tous les k entre 1 et ℓ et tous les j entre 1 et m. Le nombre total de requêtes à effectuer est donc $m \times \ell = \mathcal{O}(nd)$.

Il est donc possible de reconstruire un graphe biparti à n sommets et où le degré des noeuds dans une des deux parties est borné par une constante d en $\mathcal{O}(nd)$ requêtes par un algorithme non-adaptatif. Grâce au théorème 4, on peut transformer cet algorithme pour obtenir un autre algorithme de complexité $\mathcal{O}(nd)$ qui résoud le problème de la reconstruction de graphes à degré borné.

Avec d'autres considérations plus techniques, on démontre que $\mathcal{O}(nd)$ requêtes sont non seulement suffisantes mais aussi *nécessaires* à la reconstruction d'un graphe à degré borné par d à n sommets. On a donc le théorème suivant :

Théorème 5 *Les graphes à degré borné par une constante d peuvent être reconstruits par un algorithme non-adaptatif effectuant dans le pire des cas $\mathcal{O}(nd)$ requêtes. Cet algorithme a une complexité optimale à un facteur multiplicatif près.*

La preuve de l'existence d'un tel algorithme peut être généralisée des graphes à degré borné aux graphes k-dégénérés. On fait encore appel à la représentation bipartie des graphes, mais l'utilisation d'algorithmes pour la reconstruction de vecteurs à poids borné vient dans ce cas remplacer les matrices de d-detection et de d-séparation. La théorie de l'information

permet aussi de calculer une borne inférieure sur la complexité du problème de reconstruction de graphes k-dégénérés. Le résultat obtenu dans [9] est le suivant :

Théorème 6 *Pour tous n, k, il existe un algorithme de reconstruction pour la classe des graphes k-dégénérés à n sommets de complexité $\mathcal{O}(nk)$. Cette complexité atteind la borne inférieure fournie par la théorie de l'information à un facteur multiplicatif près.*

4.4 Circuits Hamiltoniens

La reconstruction de circuits Hamiltoniens constitue une part importante du travail de thèse de Vladimir Grebinski. Ma première tâche au Loria a été de bien comprendre ce travail, de manière à pouvoir le réintégrer dans un cadre plus global, et pour essayer d'utiliser les techniques employées sur d'autres classes de graphes. Cette partie présente un résumé des idées de Vladimir, qui sont parfois complexes, notamment en ce qui concerne l'étude dans le modèle quantitatif. Pour plus de détails, on pourra se référer au quatrième chapitre de sa thèse [9].

Rappelons brièvement que les problèmes de reconstruction de circuits Hamiltoniens, tous comme ceux de reconstruction de couplages étudiés après, sont motivés par des problèmes biologiques réels, liés au séquençage du génome. Les détails figurent en section 4.1.

4.4.1 Étude dans le modèle booléen

Comme il y a $\frac{(n-1)!}{2}$ circuits Hamiltoniens sur n sommets, la théorie de l'information permet d'obtenir la borne suivante sur la complexité de reconstruction de circuits Hamiltoniens : $\log_2 \frac{(n-1)!}{2} = \Omega(n \log_2 n)$. Cette borne inférieure peut être atteinte sous certaines conditions, comme le précise le théorème suivant :

Théorème 7 *La borne inférieure $\Omega(n \log_2 n)$ sur la complexité de la reconstruction d'un circuit Hamiltonien peut être atteinte par un algorithme* adaptatif. *Mais il ne peut pas exister d'algorithme* non-adaptatif *qui résolve ce problème avec une complexité en $\mathcal{O}(n \log_2 n)$.*

Remarquons aussi que M. Aigner démontre dans [1] que tout algorithme qui résoud le problème de reconstruction de circuit Hamiltonien dans le modèle 2-sommet a une complexité quadratique. Ce modèle est une restriction du modèle booléen, où les questions ne peuvent porter que sur les sous-ensembles de sommets de taille 2, c'est-à-dire où chaque question teste une arête possible.

Dans [10], Vladimir et Gregory ont décrit un algorithme adaptatif pour la reconstruction d'un circuit Hamiltonien H en $2n \log_2 n$ requêtes. Pour

décrire l'idée principale de cet algorithme, on a besoin de donner quelques définitions.

Définition 13 *Une* chaine c *est une suite finie de sommets* $\langle a_1, \ldots, a_t \rangle$, $t \geq 1$ *telle que pour tout* j, $1 \leq j \leq t-1$, (a_j, a_{j+1}) *est une arête de* H.
Pour toute chaine $c = \langle a_1, \ldots, a_t \rangle$, *nous définissons* $left(c) = a_1$ *et* $right(c) = a_t$.
On dit qu'un ensemble de chaines $\mathcal{C}$ *est* indépendant *si pour toutes chaines* c_1 *et* c_2 *de* $\mathcal{C}$, $(left(c_1), left(c_2))$, $(left(c_1), right(c_2))$ *et* $(right(c_1), right(c_2))$ *ne sont pas des arêtes du circuit Hamiltonien* H.

L'algorithme fonctionne par insertion des sommets un à un dans un ensemble indépendant de chaines $\mathcal{C}$. Il est important de remarquer que, lorsque l'on insère un nouveau sommet x dans l'ensemble $\mathcal{C}$, les voisins possibles de x dans le graphe H, et parmi les sommets dèjà examinés sont seulement les $left(c)$ et $right(c)$ pour $c \in \mathcal{C}$. Ainsi, l'insertion de chaque sommet peut être faite en $2 \log n$ requêtes, ce qui permet d'obtenir un algorithme adaptatif pour la reconstruction de circuits Hamiltoniens qui a une complexité de $2n \log n$ requêtes.

Il est intéressant de remarquer que cette complexité ne peut pas être atteinte pas un algorithme non-adaptatif. Dans l'article [3], les auteurs montrent que $\Omega(n^2)$ requêtes sont nécessaires, pour un algorithme non-adaptatif resolvant le problème de reconstruction de circuits Hamiltoniens. Le résultat démontré dans cet article est en fait plus général, et implique que $\Omega(n^2)$ requêtes sont nécessaires dans tout algorithme non-adaptatif reconstruisant un graphe pris dans une des classes suivantes : couplages, couplages parfaits, circuits Hamiltoniens, graphes isomorphes à un graphe à degré borné fixé, avec $\Omega(n)$ arêtes ...

Cet exemple montre bien le saut de complexité qui peut exister entre la version adaptative d'un problème et son équivalent non-adaptatif.

4.4.2 Étude dans le modèle quantitatif

Comme dans le modèle booléen, la théorie de l'information permet de dériver une borne inférieure, en tenant compte du fait que dans le modèle quantitatif, il n'y a plus 2 mais $n+1$ réponses possibles à chaque question. On a le théorème suivant :

Théorème 8 *Dans le modèle quantitatif, la borne inférieure de la théorie de l'information sur la complexité de reconstruction d'un circuit Hamiltonien est* $\log_{n+1} \frac{(n-1)!}{2} = (1 + o(1)) \cdot n$.

Dans [10], Vladimir et Gregory présentent un algorithme atteignant à un facteur multiplicatif près cette borne inférieure de complexité, pour la reconstruction d'un circuit Hamiltonien H. Il se déroule en deux étapes, la première étape étant adaptative, et la seconde non-adaptative.

Première étape : Dans cette première étape, le but est de réduire le problème de reconstruction de circuits Hamiltoniens au problème de reconstruction de graphes bipartis. Pour ce faire, on n'utilise pas la représentation bipartie du graphe, mais on partitionne les sommets de V en trois sous-ensembles $V_1 \uplus V_2 \uplus V_3$, de manière à ce qu'aucune arête ne relie deux sommets pris dans la même classe de la partition. Ceci peut être fait simplement en considérant les sommets un par un et en construisant au fur et à mesure V_1, V_2 et V_3. Chaque sommet ayant exactement deux voisins, ceci peut être fait en au plus $2n$ requêtes. On met ainsi en évidence le caractère 3-parti du circuit Hamiltonien H. La reconstruction d'un graphe 3-parti se réduit aisément à celle d'un graphe biparti, en utilisant trois fois une procédure reconstruisant un graphe biparti, pour les trois paires possibles de classes de la 3-partition des sommets du graphe.

Deuxième étape : Après la première étape, on peut s'intéresser seulement au problème de la reconstruction de graphes bipartis. Comme les degrés des sommets sont bornés par 2 dans chaque sous-graphe de H induit par l'ensemble de sommets formé de l'union de deux des V_i, les graphes bipartis que l'on a besoin de considérer sont à degré borné dans au moins une des deux parties, et on se retrouve donc face à un problème déjà traité (preuve du théorème 5), grâce à des techniques élaborées provenant des problèmes de pesages de pièces.

L'algorithme proposé dans [10], et dont on a présenté ici la trame permet donc, en utilisant des techniques de *group testing* de démontrer le théorème suivant :

Théorème 9 *Un circuit Hamiltonien sur n sommets peut être reconstruit en $\mathcal{O}(n)$ requêtes grâce à un algorithme en deux étapes, la première étant adaptative et la seconde non-adaptative.*

4.5 Couplages et couplages parfaits

4.5.1 Étude dans le modèle booléen

Rappelons pour commencer le problème dont on traite ici. Il s'agit de la reconstruction d'un graphe pris dans la classe des couplages sur n sommets, ou des couplages parfaits sur n sommets, à l'aide de questions du type « Pour un sous-ensemble de sommets $Q \subseteq V$, existe-t-il au moins une arête du couplage reliant deux sommets de Q ? ».

L'étude de la reconstruction de couplages et de couplages parfaits dans le modèle booléen a été commencée dans un article de N. Alon et al. [3], qui considère seulement le point de vue des algorithmes non-adaptatifs (c'est-à-dire des ensembles de questions indépendantes les unes des autres) qui reconstruisent un couplage. On mentionne ici un résultat obtenu dans cet

article, qui concerne la complexité de reconstruction d'un couplage par un algorithme non-adaptatif, avant d'étudier cette complexité pour les algorithmes adaptatifs. Sur l'exemple des couplages, on se rend compte de la puissance apportée par l'adaptativité des algorithmes : le gain en complexité est énorme. En effet, la meilleure complexité que l'on peut espérer pour un algorithme non-adaptatif est quadratique (c'est-à-dire guère mieux que l'algorithme naïf qui consiste à tester toutes les arêtes possibles) alors que l'on atteind à un facteur multiplicatif près la borne inférieure de la théorie de l'information par un algorithme adaptatif simple.

Algorithmes non-adaptatifs Dans [3], les auteurs prouvent le théorème suivant :

Théorème 10 *Pour tout entier $n \geq 3$, tout algorithme non-adaptatif reconstruisant un couplage sur n sommets effectue au moins $\frac{49}{153}\binom{n}{2}$ requêtes.*

Une extension de ce résultat, qui est aussi énoncée et prouvée dans le même article, a en particulier pour conséquence que :

Théorème 11 *$\Omega(n^2)$ requêtes sont nécessaires à tout algorithme non-adaptatif qui reconstruit un couplage parfait sur n sommets.*

Borne inférieure sur la complexité L'énumération des couplages est un problème ouvert, et l'utilisation de la théorie de l'information pour le cacul d'une borne inférieure, ce qui demande de connaître cette énumération, semble donc prohibée. Cependant, il est simple d'énumérer les couplages parfaits, et leur énumération fournit une borne inférieure sur le nombre de couplages généraux. Le théorème suivant est prouvé en annexe :

Théorème 12 *Le nombre de couplages parfaits sur n sommets est $PM_n = \frac{n!}{2^{\lfloor \frac{n}{2} \rfloor} \times \lfloor \frac{n}{2} \rfloor !}$. Ainsi, le nombre M_n de couplages généraux sur n sommets est supérieur à $\frac{n!}{2^{\lfloor \frac{n}{2} \rfloor} \times \lfloor \frac{n}{2} \rfloor !}$.*

Dans le modèle booléen, ce résultat permet de calculer la borne inférieure suivante par la méthode de la théorie de l'information :

Théorème 13 *$\log_2(PM_n) = (1 + o(1))(\frac{n}{2} \log_2 n)$ requêtes sont nécessaires pour reconstruire un couplage parfait sur n sommets. Ceci est aussi valable pour les couplages généraux sur n sommets.*

Algorithme adaptatif Contrairement à ce qu'on peut observer dans le cas d'algorithmes non-adaptatifs, la borne inférieure du paragraphe précédent peut être atteinte par un algorithme adaptatif. On propose dans la suite un algorithme qui reconstruit un couplage parfait sur n sommets en $(1 + o(1))(\frac{n}{2} \log_2 n)$ requêtes, ce qui est précisement la borne calculée avant.

Cet algorithme s'étend aux couplages généraux, mais sa complexité est dans ce cas supérieure d'un facteur multiplicatif 2 à cette borne inférieure.

L'algorithme dont il est question est composé de deux étapes. La première est adaptative : elle vise à partitionner l'ensemble des sommets en deux sous-ensembles S_1 et S_2 de manière à ce que deux sommets du même sous-ensemble ne soient jamais reliés par une arête du couplage cherché. Au cours de la seconde étape, qui au contraire peut-être realisée de manière non-adaptative, on va retrouver les arcs du couplage entre S_1 et S_2.

Les détails de l'algorithme et l'évaluation de sa complexité figurent en annexe.

On peut résumer les résultats obtenus pour les couplages et les couplages parfaits dans le théorème suivant :

Théorème 14 *Il existe un algorithme adaptatif permettant de reconstruire un couplage parfait sur n sommets en $(\frac{1}{2} + o(1))(n \log_2 n)$ requêtes, ce qui est exactement la borne inférieure fournie par la théorie de l'information. Il existe un algorithme adaptatif permettant de reconstruire un couplage général sur n sommets en $(1 + o(1))(n \log_2 n)$ requêtes, ce qui est à un facteur 2 près la borne inférieure fournie par la théorie de l'information.*

4.5.2 Étude dans le modèle quantitatif

On considère maintenant le problème de la reconstruction de couplages, non plus le modèle *boléen* comme avant, mais le modèle *quantitatif*. On rappelle que les questions sont alors du type : « Pour un sous-ensemble de sommets $Q \subseteq V$, combien existe-t-il d'arêtes du couplage reliant deux sommets de Q ? ». On propose d'abord une borne inférieure pour la complexité de ce problème, puis on explique brièvement comment construire un algorithme qui atteind cette complexité à un facteur multiplicatif près. Remarquons que l'étude du cas des couplages parfaits donne lieu aux mêmes résultats par l'utilisation des mêmes techinques.

Borne inférieure de la complexité Le nombre d'arêtes dans un couplage sur n sommets est borné par $\lfloor \frac{n}{2} \rfloor$. Le nombre de réponses possibles à une question $Q \subseteq V$ est donc $\lfloor \frac{n}{2} \rfloor + 1$. On utilise comme avant la méthode de calcul de borne inférieure par la théorie de l'information, grâce à la minoration du nombre de couplages calculée plus haut : $M_n \geq \frac{n!}{2^{\lfloor \frac{n}{2} \rfloor} \times \lfloor \frac{n}{2} \rfloor !}$.

On obtient ainsi le théorème suivant :

Théorème 15 *Au moins* $(1+o(1))\Big(\log_{\lfloor \frac{n}{2} \rfloor + 1} \big(\frac{n!}{2^{\lfloor \frac{n}{2} \rfloor} \times \lfloor \frac{n}{2} \rfloor !} \big) \Big) = (1+o(1))(\frac{n}{2})$ *requêtes sont nécessaires à tout algorithme reconstruisant un couplage sur n sommets dans le modèle quantitatif.*

Description d'un algorithme optimal à un facteur multiplicatif près
Il suffit en fait de remarquer qu'un couplage est un graphe biparti avec degré borné par 1 et d'utiliser l'algorithme proposé par Vladimir Grebinski (voir section 4.3) pour la reconstruction de graphes bipartis où les degrés des sommets appartenant à un des sous-ensembles sont bornés par d. Dans une première étape, on explicite le caractère biparti du graphe en trouvant deux sous-ensembles de sommets S_1 et S_2 comme dans le cas du modèle booléen. Puis on utilise l'algorithme de Vladimir qui a pour complexité $8dn + o(n)$ dans le cas général, ce qui dans notre cas donne une complexité linéaire pour la reconstruction de couplages.

4.6 Étoiles et cliques

Rappelons les définitions données auparavant et qui vont nous intéresser particulièrement ici. On note S_k l'ensemble de tous les graphes constitués d'une copie d'une étoile à $k+1$ sommets et de $n-k-1$ sommets isolés. On note aussi C_k l'ensemble des graphes composés d'une clique de taille k et de $n-k$ sommets isolés. On va ici considérer les problèmes de reconstruction de graphes pris dans une des classes $\mathit{Stars} = \cup_{k=0}^{n-1} S_k$ et $\mathit{Cliques} = \cup_{k=1}^{n} C_k$.

4.6.1 Étude dans le modèle booléen

Noga Alon et Vera Asodi ont étudié le problème de la reconstruction, par un algorithme non-adaptatif, de cliques et d'étoiles dans le modèle booléen. Parmi les résultats exposés dans [2], on retiendra ici les suivants :

Théorème 16 *Le nombre de requêtes faites par un algorithme non-adaptatif reconstruisant un graphe de Stars est borné inférieurement par* $\binom{n}{2}$. *Cette borne inférieure peut être atteinte en testant naïvement toutes les paires de sommets.*

Théorème 17 *Tout algorithme non-adaptatif reconstruisant un graphe de Cliques fait au moins* $\Omega(n \log n)$ *requêtes. Et il existe un algorithme non-adaptatif résolvant ce problème en* $O(n \log^2 n)$ *requêtes.*

Aucun résultat concernant les algorithmes adaptatifs n'existait en revanche dans la littérature. J'ai donc repris l'étude du problème à la base. Pour le modèle booléen, dans le cas des étoiles comme dans celui des cliques, la théorie de l'information fournit des bornes inférieures linéaires. On peut construire des algorithmes à deux tours ayant une complexité optimale à un facteur multiplicatif deux près. Ceci contraste avec les résultats présentés dans [2] pour les algorithmes en une étape. On peut aussi concevoir des algorithmes purement adaptatifs, mais qui ont une complexité encore meilleure que les précédents, très proche de l'optimum.

Bornes inférieures La technique utilisée ici pour trouver des bornes inférieures sur les complexités de reconstruction de cliques ou d'étoiles est simplement celle de la théorie de l'information. Les bornes inférieures fournies pourront être atteintes à un facteur multiplicatif près, ce qui n'incite pas à en rechercher de meilleures par des arguments plus puissants.

Étoiles Pour calculer le nombre de graphes présents dans *Stars*, qui nous donnera accès à la borne inférieure de la théorie de l'information, commençons par évaluer la cardinalité de S_k.

Il y a n choix possibles pour le centre de l'étoile, et on choisit ensuite les k sommets de la couronne parmi les $n-1$ restants pour déterminer complètement une étoile de S_k. On a ainsi :

$$|S_k| = n \times \binom{n-1}{k}$$

En considérant le graphe sans aucune arête comme une étoile possible, et en étant attentif au fait que dans une étoile composée d'une arête seulement, la définition de centre est ambigüe, on obtient pour la cardinalité de *Stars* le résultat suivant :

$$\begin{aligned} |Stars| &= n \times \sum_{k=2}^{n-1} \binom{n-1}{k} + \frac{n(n-1)}{2} + 1 \\ &= n \times (2^{n-1} - 1) - \frac{n(n-1)}{2} + 1 \end{aligned}$$

Travaillant dans le modèle booléen, la borne inférieure de la théorie de l'information est $\log_2 |Stars| = (1 + o(1))n$.

Cliques Pour définir une clique de k sommets sur n sommets, il faut et il suffit de choisir quels sont les k sommets parmi les n possibles qui font partie de la clique. Ainsi, il est clair que $|C_k| = \binom{n}{k}$.

Par conséquent,

$$|Cliques| = \sum_{k=0}^{n} \binom{n}{k} = 2^n$$

Et donc dans le modèle booléen, la borne inférieure sur le nombre de requêtes fournie par la théorie de l'information est encore linéaire : $\log_2 |Cliques| = (1 + o(1))n$ requêtes sont nécessaires pour la reconstruction d'une clique de *Cliques*.

On peut résumer ces résultats en un théorème :

Théorème 18 *Une borne inférieure sur la complexité de la reconstruction d'étoiles ou de cliques à n sommets dans le modèle booléen est $(1 + o(1))n$ requêtes.*

Principe général des algorithmes à deux tours présentés Pour reconstruire les étoiles comme les cliques, la structure de l'algorithme à deux tours proposé sera la suivante :

- Au cours du premier tour, on va rechercher un point de départ pour amorcer la reconstruction : il s'agit du centre de l'étoile dans le premier cas, et d'un point quelconque de la clique dans le second.
- Dans le deuxième tour, on va utiliser l'information obtenue avant pour reconstruire le reste du graphe.

Pour rechercher le centre d'une étoile sur n sommets, dont éventuellement certains sont isolés, il suffit de faire la remarque suivante.

Considérons d'abord le cas où les étoiles sont composées d'au moins trois sommets. Alors les requêtes $V \setminus \{i\}$ pour $1 \leq i \leq n$ donnent toutes la réponse `vrai`, sauf une. Le sommet i correspondant est le centre de l'étoile.

Pour le cas de l'étoile réduite à une arête, deux requêtes exactement produisent la réponse `faux` : celles correspondant aux deux extrémités de l'arête. On choisit pour jouer le rôle de centre n'importe lequel de ces deux sommets.

Enfin le cas de l'étoile vide correspond au cas où toutes les réponses obtenues sont négatives. [3]

Connaissant donc le centre x de l'étoile à l'issue d'un premier tour où les requêtes effectuées sont les $V \setminus \{i\}$ pour $1 \leq i \leq n$, il est facile de reconstruire le reste du graphe grâce aux requêtes $\{x, i\}$ pour $1 \leq i \leq n$, $i \neq x$.

Pour retrouver un point d'une clique, on peut appliquer l'idée suivante. Considérons les requêtes $Q_i = \{1, \ldots, i\}$ pour $2 \leq i \leq n$. Il existe un sommet numéro i_0 tel que les réponses à $Q_2, \ldots, Q_{i_0-1}$ sont `faux` et les réponses à $Q_{i_0}, \ldots, Q_n$ sont `vrai` (sauf dans le cas de la clique vide où toutes les réponses sont `faux`). Le sommet i_0 fait alors partie de la clique et constitue un point de départ pour la reconstruction du reste du graphe.

Il suffit ensuite de constater qu'un point est dans la clique si et seulement s'il est voisin de i_0. Dans le deuxième tour, on peut donc simplement utiliser les requêtes $\{i_0, i\}$ pour $1 \leq i \leq n$, $i \neq i_0$.

Les détails des algorithmes figurent en annexe.

Théorème 19 *Cliques et étoiles peuvent être reconstruite en* $2n + o(n)$ *requêtes par un algorithme à deux tours.*

Accélération avec des algorithmes purement adaptatifs Les algorithmes présentés ci-dessus sont optimaux à un facteur deux près. En outre,

[3] Ce qui bien sur ne permet pas de distinguer les deux "étoiles" possibles sur deux sommets. On supposera donc que $n \geq 3$

ils ont l'avantage d'être d'une conception simple et à deux tours seulement. Cependant, il est possible de rendre leur complexité encore plus proche de la borne inférieure fournie par la théorie de l'information, mais au détriment de ces propriétés.

La structure globale des algorithmes décrits dans la suite est globalement identique à celle exposée plus haut : recherche d'un point de départ puis reconstruction du reste du graphe. Il est à noter cependant que les algorithmes de cette section sont *purement adaptatifs*. Pour être plus précis, ils sont composés de plusieurs étapes dont certaines sont purement adaptatives.

Dans le cas des étoiles, on proposera un algorithme en trois étapes. La première met en évidence une arête de l'étoile s'il y en a une par une recherche dichotomique. Ceci permet de trouver dans une deuxième étape le centre de l'étoile, puis au cours de la dernière étape de reconstruire la totalité du graphe.

Le cas des cliques est plus simple : une première étape purement adaptative permet de trouver un point de la clique, et une seconde de la reconstruire entièrement.

On pourra trouver en annexe une description plus précise des algorithmes évoqués.

Théorème 20 *Cliques et étoiles peuvent être reconstruite en $n+o(n)$ requêtes par un algorithme purement adaptatif.*

4.6.2 Étude dans le modèle quantitatif

Bornes inférieures On rappelle les cardinalités de *Stars* et de *Cliques* calculées avant :

$$\mathit{Stars} = n \times (2^{n-1} - 1) - \frac{n(n-1)}{2} + 1 \quad \text{et} \quad \mathit{Cliques} = 2^n$$

On rappelle aussi que dans le modèle quantitatif, les questions sont du type : " Pour $Q \subseteq V$, combien y a-t-il d'arête(s) du graphe caché reliant deux sommets de Q ? " Les réponses obtenues sont donc des nombres entre 0 et $n-1$ dans le cas des étoiles, entre 0 et $\frac{n(n-1)}{2}$ dans le cas des cliques. Ainsi les bornes inférieures sur le nombre de requêtes faites par un algorithme résolvant dans le modèle quantitatif les problèmes de reconstruction d'étoiles et de cliques respectivement sont :

$$\log_n \left(n \times (2^{n-1} - 1) - \frac{n(n-1)}{2} + 1 \right) = \Omega(\frac{n}{\log_2 n})$$

$$\text{et} \quad \log_{\frac{n(n-1)}{2}+1}(2^n) = \Omega(\frac{n}{2\log_2 n})$$

Structure des algorithmes Comme dans le cas du modèle booléen, on va proposer ici des algorithmes purement adaptatifs fonctionnant par étapes, qui renconstruisent des étoiles ou des cliques dans le modèle quantitatif. Ces algorithmes seront d'une complexité asymptotiquement optimale à un facteur multiplicatif près.

La structure globale des algorithmes utilisés dans le modèle booléen peut être conservée. En outre, toute requête dans le modèle booléen peut être très facilement simulée par une requête dans le modèle quantitatif. Ainsi, il est possible en deux étapes, faisant un nombre logarithmique de requêtes quantitatives au total, de retrouver le centre d'une étoile cachée. De même, on peut en un nombre logarithmique de requêtes quantitatives exhiber un sommet appartenant à une clique que l'on cherche à reconstruire. Il suffit pour cela d'utiliser les premières étapes purement adaptatives des algorithmes pour le modèle booléen qui sont décrits en annexe.

Des modifications doivent en revanche être apportées aux dernières étapes. Remarquons simplement que dans le cas des étoiles comme dans celui des cliques, la dernière étape consiste à rechercher les voisins d'un sommet, parmi les $n-1$ autres. Il suffit donc d'utiliser, pour remplacer ces dernières étapes, la représentation bipartie de ces graphes, puis un algorithme de recherche d'un nombre arbitraire de fausses pièces parmi n.

La recherche d'un nombre arbitraire de fausses pièces parmi n peut être résolue en $(2 + o(1))\frac{n}{\log_2 n}$ requêtes (voir [13] et [10]). Il est à remarquer que la construction proposée dans ces articles conduit à un algorithme non adaptatif.

Ainsi, la seule amélioration concernant la recherche d'un nombre arbitraire de fausses pièces parmi n permet, dans le cadre du modèle quantitatif, d'atteindre encore une complexité optimale à un facteur multiplicatif près pour la reconstruction d'étoiles ou de cliques par un algorithme adaptatif : en $O(\frac{n}{\log_2 n})$.

Perspectives Dans le cadre du modèle quantitatif, on n'a pas envisagé d'algorithmes en deux étapes ou non adaptatifs. Il serait intéressant de raffiner la recherche de borne inférieure dans ce cadre, et de voir si on peut trouver un algorithme dont la complexité atteindrait cette borne.

5 Conclusion

Dans ce rapport de stage, j'espère avoir su présenter la recherche combinatoire de manière intelligible et sous plusieurs aspects. D'abord son côté ludique, avec des problèmes simples, inspirés de situations réelles, qui peuvent cependant faire émerger déjà des concepts théoriques importants. Ensuite, les applications à la bioinformatique de la recherche combinatoire sur les graphes, qui ne peut se faire sans une étude approfondie des problèmes de

pesage de pièces. J'espère aussi avoir su montrer la diversité des problèmes qu'on peut recontrer en recherche combinatoire, et l'hétérogénéité des résultats connus. Enfin, sur l'exemple des problèmes de reconstruction de graphes dans certaines classes données, et malgré la simplicité des énoncé des problèmes, j'espère avoir montré la diversité des outils utilisés et la complexité de leur imbrication.

La recherche combinatoire est toujours un domaine de travail riche de problèmes ouverts, et où de nouveaux problèmes ayant des applications directes peuvent apparaitre naturellement en étant attentif aux ennuis techniques auquels sont confrontés des collègues biologistes par exemple. J'ai pris conscience pendant mon stage de l'étendue de ce domaine, et apporté ma petite pierre à l'édifice. Cependant je laisse en suspens des points que j'aurais aimé approfondir, notamment en ce qui concerne la reconstruction d'étoiles et de cliques. Une prochaine fois peut-être ...

Annexe

Preuve du théorème 12

Pour les n pairs, on démontre par récurrence que le nombre de couplages parfaits sur n sommets est $PM_n = \frac{n!}{2^{\frac{n}{2}}(\frac{n}{2})!}$.

- Initialisation :
  pour $n = 2$, il y a un unique couplage parfait, et on a bien $PM_2 = 1$.
- Hérédité :
  supposons qu'il y ait $PM_{(n-2)}$ couplages parfaits sur K_{n-2}. Un couplage parfait sur K_n peut être décrit de la façon suivante : un arc associe le sommet numéro 1 à un sommet j choisi parmi $\{2, 3, \ldots, n\}$, et le reste des sommets forme un couplage sur K_{n-2}, quitte à renuméroter les sommets de $\{1, 2, \ldots, n\} \setminus \{1, j\}$ sur $\{1, 2, \ldots, n-2\}$ en respectant l'ordre entre les sommets. Ainsi, le nombre de couplages parfaits sur K_n est

$$\begin{aligned}
(n-1) \times PM_{n-2} &= \frac{(n-1) \times (n-2)!}{2^{\frac{n-2}{2}}(\frac{n-2}{2})!} \\
&= \frac{n!}{n \times 2^{\frac{n-2}{2}}(\frac{n-2}{2})!} \\
&= \frac{n!}{2 \times 2^{\frac{n-2}{2}} \times \frac{n}{2} \times (\frac{n-2}{2})!} \\
&= \frac{n!}{2^{\frac{n}{2}}(\frac{n}{2})!} = PM_n
\end{aligned}$$

Nous avons donc démontré la propriété voulue par récurrence sur n, n pair. Dans le cas où n est impair, on obtient facilement, en ajoutant un sommet

isolé a un couplage parfait sur $n-1$ sommets, que le nombre de couplages sur n sommets est aussi borné inférieurement par $\frac{n!}{2^{\frac{n-1}{2}} \times (\frac{n-1}{2})!}$. Ainsi, en notant M_n le nombre de complages à n sommets, on a, pour tout n :

$$M_n \geq \frac{n!}{2^{\lfloor \frac{n}{2} \rfloor} \times \lfloor \frac{n}{2} \rfloor !}$$

Algorithme adaptatif pour la reconstruction de couplages et de couplages parfaits dans le modèle booléen

Pour les couplages parfaits d'abord, on propose l'algorithme suivant :
Etape 1 :
$S_1 \leftarrow \emptyset$ et $S_2 \leftarrow \emptyset$
Pour i allant de 1 à n, faire
si la réponse à $\{i\} \cup S_1$ est `oui`
alors $S_2 \leftarrow \{i\} \cup S_2$
sinon $S_1 \leftarrow \{i\} \cup S_1$
fin pour
Retourner S_1 et S_2

Le nombre de requêtes effectuées à cette première étape de l'algorithme est n. Comme tout sommet a exactement un voisin dans le couplage, cet algorithme est correct et à la fin on recupère deux ensembles S_1 et S_2 de cardinalité chacun $\frac{n}{2}$.

Etape 2 :
A cette étape, on va, pour chaque sommet de S_2, rechercher son unique voisin dans S_1. Par recherche dichotomique, le coût total serait donc $|S_2| \times \log_2 |S_1| = \frac{n}{2} \times \log_2 \frac{n}{2}$. Pour effectuer cette étape de manière non-adaptative, on peut utiliser $\frac{n}{2}$ fois un algorithme non-adptatif de recherche d'une fausse pièce parmi $\frac{n}{2}$, ce qui demande le même nombre de requêtes.

L'algorithme précedent peut être utilisé tel quel pour la recherche d'un couplage qui n'est pas nécessairement parfait. C'est l'évaluation de sa complexité qui change. Notons p le nombre d'arcs du couplage cherché. Alors à la fin de la première étape de l'algorithme, nous obtenons des ensembles S_1 et S_2 de cardinalités respectives $n-p$ et p. On conclut donc que l'algorithme proposé pour la reconstruction d'un couplage parfait sur K_n appliqué au problème de reconstruction d'un couplage quelconque à p arêtes sur K_n fait $(1+o(1))(p \log_2(n-p))$ requêtes. Comme p n'est pas connu à l'avance pour les couplages généraux, on obtient donc un algorithme adaptatif reconstruisant un couplage avec au plus $(1+o(1))(n \log_2 n)$ requêtes.

Algorithme à deux tours pour la reconstruction d'étoiles dans le modèle booléen

Premier tour Dans cette étape, on repère que l'étoile cherchée est vide si c'est le cas, et sinon, on retourne une variable x qui contient un sommet : le sommet au centre de l'étoile.

1. Pour tout $i \in V$ tester $V \setminus \{i\}$
2. Si toutes les réponses sont `faux`, alors l'étoile cherchée est l'étoile vide et on s'arrête.
3. Si exactement deux réponses sont `faux`, disons les réponses à $V \setminus \{i_1\}$ et $V \setminus \{i_2\}$, alors $x \leftarrow i_1$
4. Sinon, exactement une réponse est `faux`, disons la reponse à $V \setminus \{i_1\}$, et alors $x \leftarrow i_1$
5. Retourner x

Deuxième tour Ayant trouvé le centre, il ne reste plus alors qu'à tester toutes les arêtes potentielles, qui sont celles faisant intervenir le centre.

1. $S \leftarrow V \setminus \{x\}$
2. Pour tout $i \in S$ tester $\{x, i\}$

Complexité Le nombre de requêtes effectuées est n pour le premier tour, et $n-1$ pour le second, soit un total de $2n-1$ requêtes. Ce nombre de requêtes est seulement le double du nombre minimum de requêtes que l'on pouvait espérer après le calcul de la borne inférieure par des arguments de théorie de l'information.

Algorithme à deux tours pour la reconstruction de cliques dans le modèle booléen

Premier tour Dans cette étape, on repère que la clique cherchée est vide si c'est le cas, et sinon, on retourne une variable v qui contient un sommet appartenant à la clique.

1. Pour tout $i \in V \setminus \{1\}$ tester $Q_i = \{1, \ldots, i\}$
2. Trouver le plus petit indice v tel que Q_v répond `vrai`, et s'il n'y en a pas, on a trouvé que la clique recherchée est la clique vide et on s'arrête.
3. Retourner v

Deuxième tour Un sommet de la clique mis en évidence, il ne reste plus qu'à trouver ses voisins.

1. $S \leftarrow V \setminus \{v\}$
2. Pour tout $i \in S$ tester $\{v, i\}$

Complexité Le nombre de requêtes effectuées ici est $2n-2$, soit le double de la borne inférieure calculée plus haut.

Algorithme purement adaptatif pour la reconstruction de cliques dans le modèle booléen

Première étape Il s'agit ici de trouver un point appartenant à la clique. Pour cela, on procède en deux temps. Tout d'abord, on construit par divisions successives deux ensembles S_1 et S_2, non vides, ne contenant chacun aucune arête, mais tels qu'il existe au moins une arête entre S_1 et S_2. Ensuite, il ne reste plus qu'à trouver dans S_1 un sommet v qui est une extremité d'une arête allant vers S_2. Pour le trouver, on procède encore par dichotomie.

1. Tester V. Si la réponse est `faux`, alors on a trouvé la clique sans arête et on s'arrête. Si la réponse est `vrai`, alors :
2. $S \leftarrow V$
3. Partitionner S en $S_1 \uplus S_2$, avec $\big||S_1| - |S_2|\big| \leq 1$
4. Tester S_1
5. Si la réponse est `vrai` alors $S \leftarrow S_1$ et retourner en 3
6. Si la réponse est `faux` alors tester S_2
7. Si la réponse est `vrai` alors $S \leftarrow S_2$ et retourner en 3
8. Si la réponse est `faux` alors
9. $T \leftarrow S_1$
10. Partitionner T en $T_1 \uplus T_2$, avec $\big||T_1| - |T_2|\big| \leq 1$
11. Tester $T_1 \cup S_2$
12. Si la réponse est `vrai` alors
13. Si $|T_1| = 1$, alors noter $T_1 = \{v\}$ et aller en 18
14. Sinon, $T \leftarrow T_1$ et retourner en 10
15. Si la réponse est `faux` alors
16. Si $|T_2| = 1$, alors noter $T_2 = \{v\}$ et aller en 18
17. Sinon, $T \leftarrow T_2$ et retourner en 10
18. Retourner v

Cette première étape est purement adaptative, et le nombre de requêtes effectuées est logarithmique. En effet, les itérations du processus de division sont au nombre de $\lceil \log_2 n \rceil$ au plus, et la cardinalité de S_1 est bornée par $\lceil \frac{n}{2} \rceil$. Ainsi, le nombre de requêtes effectuées par l'algorithme ci-dessus est borné supérieurement par :

$$2\lceil \log_2 n \rceil + \left\lceil \log_2 \left\lceil \frac{n}{2} \right\rceil \right\rceil = O(\log_2 n)$$

Seconde étape Cette deuxième étape où l'on reconstruit la clique à partir du point v trouvé dans l'étape précedente se déroule exactement comme dans l'algorithme à deux tours :

1. $S \leftarrow V \setminus \{v\}$
2. pour tout $i \in S$ tester $\{v, i\}$

Cette étape demande $n - 1$ requêtes.

Complexité La complexité globale de cet algorithme purement adaptatif est donc optimale : $(1 + o(1))n$.

Algorithme purement adaptatif pour la reconstruction d'étoiles dans le modèle booléen

Première étape Le but est ici de trouver une arête de l'étoile cachée s'il en existe une. L'algorithme suivant prend en entrée un ensemble de n sommets V et exhibe une arête lorsque cela est possible. Sinon, il indique que l'étoile recherchée est celle ne contenant aucune arête.

Comme dans le cas de cliques, on construit par divisions successives deux ensembles S_1 et S_2, non vides, ne contenant chacun aucune arête, mais tels qu'il existe au moins une arête entre S_1 et S_2. Ensuite on trouve un point v dans S_1 qui possède un voisin dans S_2. Enfin, on trouve le voisin w de v dans S_2.

1. Tester V. Si la réponse est `faux`, alors on a trouvé l'étoile sans arête et on s'arrête. Si la réponse est `vrai`, alors :
2. $S \leftarrow V$
3. Partitionner S en $S_1 \uplus S_2$, avec $\big||S_1| - |S_2|\big| \leq 1$
4. Tester S_1
5. Si la réponse est `vrai` alors $S \leftarrow S_1$ et retourner en 3
6. Si la réponse est `faux` alors tester S_2
7. Si la réponse est `vrai` alors $S \leftarrow S_2$ et retourner en 3
8. Si la réponse est `faux` alors
9. $T \leftarrow S_1$
10. Partitionner T en $T_1 \uplus T_2$, avec $\big||T_1| - |T_2|\big| \leq 1$
11. Tester $T_1 \cup S_2$
12. Si la réponse est `vrai` alors
13. Si $|T_1| = 1$, alors noter $T_1 = \{v\}$ et aller en 18
14. Sinon, $T \leftarrow T_1$ et retourner en 10
15. Si la réponse est `faux` alors
16. Si $|T_2| = 1$, alors noter $T_2 = \{v\}$ et aller en 18

17. Sinon, $T \leftarrow T_2$ et retourner en 10
18. $U \leftarrow S_2$
19. Partitionner U en $U_1 \uplus U_2$, avec $\big||U_1| - |U_2|\big| \leq 1$
20. Tester $U_1 \cup \{v\}$
21. Si la réponse est `vrai` alors
22. Si $|U_1| = 1$, alors noter $U_1 = \{w\}$ et aller en 27
23. Sinon, $U \leftarrow U_1$ et retourner en 19
24. Si la réponse est `faux` alors
25. Si $|U_2| = 1$, alors noter $U_2 = \{w\}$ et aller en 27
26. Sinon, $U \leftarrow U_2$ et retourner en 19
27. Retourner $\{v, w\}$

L'analyse de la complexité de cette étape purement adaptative s'effectue comme dans le cas des cliques et on obtient un nombre de requêtes borné supérieurement par :

$$2\lceil \log_2 n \rceil + 2\left\lceil \log_2 \left\lceil \frac{n}{2} \right\rceil \right\rceil = O(\log_2 n)$$

Deuxième étape Connaissant une arête $\{v, w\}$ de l'étoile cachée, on cherche maintenant le centre de cette étoile. On sait que ce centre est soit v soit w. On expose dans la suite un algorithme pour la recherche du centre.

Son principe de fonctionnement est le suivant : il cherche dans l'ensemble des sommets différents de v et de w un " témoin " démontrant que v est le centre, i.e. qu'il cherche à démontrer l'existence d'un voisin de v autre que w.

Comme le centre est soit v soit w, on déduit qu'il n'y a aucune arête dans $S \setminus \{v, w\}$. Ainsi, il y a une arête dans $S \setminus \{w\}$ ssi v est le centre.

1. Tester $S \setminus \{w\}$
2. Si la réponse est `vrai`, alors $x \leftarrow v$
3. Si la réponse est `faux`, alors $x \leftarrow w$
4. Retourner x

On fait seulement une requête ici : cette étape de l'algorithme est en temps constant.

Troisième étape Une fois le centre x de l'étoile trouvé, il suffit de tester toutes les arêtes possibles partant de x pour reconstruire entièrement l'étoile cachée, comme dans l'algorithme à deux tours.

1. $S \leftarrow V \setminus \{x\}$
2. pour tout $i \in S$ tester $\{x, i\}$

Cette étape demande $n - 1$ requêtes.

Complexité L'algorithme purement adaptatif composé de la succession de ces trois étapes a une complexité totale optimale : $(1 + o(1))n$.

Références

[1] M. Aigner. *Combinatorial Search*. John Wiley and Sons, 1988.

[2] N. Alon and V. Asodi. Learning a hidden subgraph. In *Automata, Languages and Programming : 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, volume 3142 of *Lecture Notes in Computer Science*, pages 110–121. Springer, 2004.

[3] N. Alon, R. Beigel, S. Kasif, S. Rudich, and B. Sudakov. Learning a hidden matching : Combinatorial identification of hidden matchings with applications to whole genome sequencing. In *FOCS : IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.

[4] N. Alon, J. H. Spencer, and P. Erdös. *The probabilistic method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1992.

[5] I. Althöfer and E. Triesch. Edge search in graphs and hypergraphs of bounded rank. *Discrete Math.*, 115(1-3) :1–9, 1993.

[6] P. Damaschke. A tight upper bound for group testing in graphs. *Discrete Appl. Math.*, 48 :101–109, 1994.

[7] D. Du and F. Hwang. *Combinatorial group testing and its applications*, volume 3. Series on applied Mathematics, 1993.

[8] V. Grebinski. On the power of additive combinatorial search model. In *COCOON : Annual International Conference on Computing and Combinatorics*, 1998.

[9] V. Grebinski. *Recherche Combinatoire : Problèmes de Pesage, Reconstruction de Graphes et Applications*. PhD thesis, Mémoire de Doctorat de l'université Henri Poincaré – Nancy 1, 1998.

[10] V. Grebinski and G. Kucherov. Optimal query bounds for reconstructing a hamiltonian cycle in complete graphs. In *Proceedings of the 5th Israel Symposium on Theory of Computing and Systems, ISTCS'97 (Ramat-Gan, Israel, June 17-19, 1997)*, pages 166–173, Los Alamitos-Washington-Brussels-Tokyo, 1997. IEEE Computer Society Press.

[11] V. Grebinski and G. Kucherov. Reconstructing set partitions. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'99 (Baltimore, Maryland, January 17-19, 1999)*, pages 915–916, New York, Society for Industrial and Applied Mathematics : Philadelphia, PA, 1999. ACM SIGACT, SIAM, Association for Computing Machinery.

[12] V. Grebinski and G. Kucherov. Optimal reconstruction of graphs under the additive model. *ALGRTHMICA : Algorithmica*, 28, 2000.

[13] B. Lindström. Determination of two vectors from the sum. *J. Comb. Theory*, 6 :402–407, 1969.

[14] B. Lindström. On möbius functions and a problem in combinatorial number theory. *Canad. Math. Bull.*, 1971.

[15] B. Linström. Determining subsets by unramified experiments. In editor J.N. Srivastava, editor, *A Survey of Statistical Designs and Linear Models*, pages 407–418. North Holland, Amsterdam, 1975.