INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Test de propriété pour l'homomorphisme de graphes et d'hypergraphes

Nicolas Dalsass

N° N.A (en cours)

Mars 2011

ISSN 0249-6399 ISRN INRIA/RR--N.A (en cours)--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SOPHIA ANTIPOLIS - MÉDITERRANÉE

Test de propriété pour l'homomorphisme de graphes et d'hypergraphes

Nicolas Dalsass *

Thème : Knowledge and Data Representation and Management
Équipes-Projets GraphIK

Rapport de recherche n° N.A (en cours) — Mars 2011 — 23 pages

Résumé : Dans ce rapport, nous nous intéressons à l'utilisation de testeurs de propriétés pour la recherche d'homomorphismes de graphes et d'hypergraphes. Après avoir montré quel genre de résultat nous pouvions attendre, nous étudions comment adapter les meilleurs testeurs connus à ce jour pour la résolution du problème k-max-CSP, les limites de ces algorithmes utilisés dans le cadre de l'homomorphisme de graphes, et donnons de nouvelles bornes inférieures théoriques concernant ce genre d'algorithmes. En particulier, nous montrons que l'homomorphisme de graphe n'est pas efficacement testable dans les modèles non-dense, ce que nous prouvons en généralisant le concept de graphe de Behrend.

Mots-clés : Homomorphisme, graphe, hypergraphe, test de propriété, algorithme probabiliste

* Encadré par Jean-François Baget

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Graph and Hypergraph Homomorphism

Abstract: In this report, we study how can one use property testing algorithms in order to find homomorphisms between graphs. We first analyse what kind of results we can expect. We then study an adapatation of the best approximation algorithms designed for the k-max-CSP problems, and in what measure we can effectively use those. We also give new theoretical lower bounds for property testers for graph homomorphism testing. Specifically, we show that graph homomorphism is not efficiently testable in non-dense models. We prove that using a construction based on Behrend graphs.

Key-words: Homomorphism, graph, hypergraph, property testing, probabilistic algorithm

Table des matières

1 Cadre de travail

1.1 Les nouveaux défis de la représentation de connaissances

Sous diverses formes, l'interrogation de bases de connaissances est plus que jamais un problème d'actualité. Le problème fondamental, qui est de chercher les réponses à une requête conjonctive est en lui-même un problème NP-complet ([10]), qui s'étend en des problèmes plus difficiles encore (par exemple en présence d'ontologies, ou pour prendre en compte la négation). Aujourd'hui, nous constatons une augmentation rapide de la taille des bases de données.

Face à cette évolution, des solutions efficaces (à l'échelle industrielle) ont été trouvées, notamment sous formes de bases de données, profitant du fait que la masse de données traitées soit très structurée : ainsi l'utilisation de systèmes de clefs et d'index a permis de ramener les temps de calcul à des durées raisonnables.

Mais déjà se présentent de nouveaux enjeux : avec le développement du web (notamment du web sémantique) sont générées de grandes masses de données peu structurées (on parle généralement de données semi-structurées). Afin de pouvoir continuer à traiter le problème de la déduction, la solution que nous avons choisie est de représenter la connaissance sous forme de graphes, a la manière de [22]. Dans ce formalisme, le problème de la déduction dans le fragment existentiel positif de la logique du premier ordre peut être vu comme un problème d'homomorphisme de graphes, ou plus généralement d'hypergraphes, étiquetés et orientés ([18]).

Ainsi, nous pouvons envisager de nouvelles approches plus dynamiques (Backtrack filtré dynamiquement, structures de données adaptatives pour les bases de graphes...). Cependant, il se pourrait que ces approches se révèlent tout de même insuffisantes. A ce moment là, on peut relâcher le problème de la requête en un problème de "recherche approximative" : dans ce cas, un algorithme potentiellement inexact mais rapide semble être un bon compromis. Plus précisément, nous avons choisi d'explorer le domaine des testeurs de propriété après avoir constaté que ceux-ci permettaient d'obtenir de bons résultats pour un problème proche : la k-coloration.

Bien entendu, d'autres approches auraient été envisageables : la programmation linéaire ou le "semidefinite programming" - deux formes d'optimisation convexe, les algorithmes d'apprentissage, les marches aléatoires fondées sur le couplage rapide d'une chaîne de Markov...

1.2 L'homomorphisme de graphes : un problème fondamental

Dans le contexte des graphes conceptuels, le problème de la déduction devient un problème d'homomorphisme de graphes. Plus précisément, on montre ([9]) qu'un fait (une formule logique) H se déduit d'un fait G si et seulement si il y a un homomorphisme du graphe conceptuel associé à H dans celui associé à G. De ce point de vue, le problème de l'homomorphisme est donc bien un problème central en Intelligence Artificielle et en Représentation de Connaissances. Cependant, comme pour tout problème NP-complet , le "meilleur" algorithme connu a une complexité asymptotique exponentielle en la taille de la requête [23]. De plus, il implique un préprocessing lourd, qui fait qu'on lui préfère dans la pratique les algorithmes mis en oeuvre pour résoudre les problèmes de contraintes : backtrack, descente de gradient, ... Il est d'ailleurs intéressant de noter que le problème de l'homomorphisme profite directment de toutes les avancées effectuées dans le domaine des CSP

Par ailleurs, au delà du domaine de l'IA, le problème de l'homomorphisme de graphes est un problème qui se retrouve dans une grande variété de champs d'étude : s'il peut se voir comme une généralisation des problèmes de coloration, il survient couramment en reconnaissance de formes ou en physique statistique par exemple. Et là encore, les problèmes actuels concernent de très grands graphes (images 3D, arrangements moléculaires...)

1.3 Travail effectué dans cette étude

Pendant ce stage, nous avons étudié dans quelle mesure les algorithmes classiques venant du domaines des testeurs de propriétés pouvaient être intéressants pour approximer le problème de l'homomorphisme. Nous avons donné des résultats concernant la possible utilisation de ces algorithmes selon la propriété testée. Nous avons aussi donné des bornes inférieures concernant la complexité de tels algorithmes, et nous avons traduit des algorithmes utilisés dans d'autres domaines, notamment pour la résolution du problème k-max-CSP, en algorithmes de graphes pour l'homomorphisme.

Nous avons finalement conclu qu'on ne pouvait tester efficacement ni la requête ni la base de données pour résoudre ce problème. Une question qui reste ouverte serait de savoir si l'on peut tester efficacement un graphe construit à partir de la donnée de ces deux éléments, question pour laquelle nous donnons des graphes qui montrent qu'elle sera difficile.

Commençons par quelques définitions (classiques) pour mieux comprendre ce que nous voulons construire[1], définitions que nous restreindrons pour plus de clarté au cadre des propriétés de graphes et d'hypergraphes.

2 Tester une propriété : définitions générales

Nous ne considérerons en général que des graphes et hypergraphes simples (sans boucle, et sans (hyper)arête multiple). Cette restriction peut sembler étrange, dans la mesure ou notre problème initial semble plus général. Cependant, on peut facilement passer d'un cas à l'autre ([17]) . Dans la plupart des exemples que nous considérerons, nos graphes seront également non étiquetés et non orientés, mais ce n'est jamais indispensable.

Définition 1 (Propriété) *Soit $\mathcal{H}^{(k)}$ la famille des hypergraphes k-uniformes*[2]*. Avec cette notation, la famille $\mathcal{H}^{(2)}$ représente les graphes traditionnels sans boucle.*

Une propriété est n'importe quelle famille $\mathcal{P} \subseteq \mathcal{H}^{((k))}$, qui est invariante pour l'isomorphisme.

On dit qu'un hypergraphe G vérifie $\mathcal{P}$ si $G \in \mathcal{P}$

Cette définition peut sembler surprenante. En effet, on aimerait qu'une propriété soit quelque chose comme "être connexe" ou "être 3-colorable". Un moyen simple de la comprendre est de transformer la propriété, par exemple, "être connexe" en la propriété "appartenir à la famille des hypergraphes connexes".

Pour ce qui suit, on se donne $\mathcal{P}$ une propriété, et $d : \mathcal{H}^{(k)} \times \mathcal{H}^{(k)} \to \mathbb{R}$ une distance sur $\mathcal{H}^{(k)}$.

On se donne également G, un hypergraphe k-uniforme.

1. Pour le lecteur intéressé par un aperçu plus général du test de propriété, Goldreich a récemment écrit une introduction au sujet ([15]). Un aperçu général des algorithmes et techniques couramment utilisés a par ailleurs été écrit par Ron ([20]).
2. Un hypergraphe est dit k-uniforme si toutes ses hyperarêtes sont d'arité k.

Définition 2 (ϵ-loin) *On dit que G est ϵ-loin de $\mathcal{P}$ lorsque $\min_{H \in \mathcal{P}}(d(G,H)) > \epsilon$*

Définition 3 (Testeur de propriété) *Un testeur de propriété pour $\mathcal{P}$ est un algorithme probabiliste, qui prend en entrées G et ϵ (et éventuellement p et p') et qui :*

- *accepte G avec une probabilité $p \geq \frac{2}{3}$ si $G \in \mathcal{P}$.*
- *rejette G avec probabilité $p' \geq \frac{2}{3}$ si G est ϵ-loin de $\mathcal{P}$.*

On dit que le testeur est à erreur unilatérale s'il accepte toujours G lorsque G est dans $\mathcal{P}$. Sinon, il est dit à erreur bilatérale.

*Remarque :*Lorsque l'algorithme reçoit en entrée un graphe qui n'a pas la propriété mais n'est pas non plus loin de l'avoir, le résultat importe peu. L'idée est que si l'algorithme accepte G par erreur, ce dernier sera en revanche "facile" à modifier pour lui faire vérifier effectivement $\mathcal{P}$, dans le cas inverse le résultat est juste.

Une autre manière de voir les testeurs de propriété consiste à voir en eux des algorithmes qui donnent une "approximation probable" de la valeur que l'on souhaite mesurer.

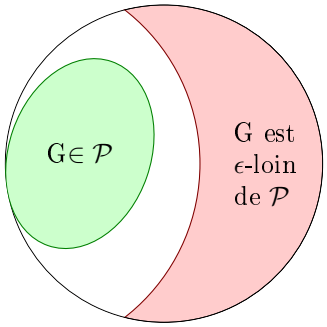

FIGURE 1 – Ensemble séparés par un testeur pour $\mathcal{P}$

Pour pouvoir mesurer l'efficacité d'un tel testeur, une mesure couramment effectuée est celle du nombre de "requêtes" qu'il effectue. Pour un testeur, faire une "requête", c'est demander si un k-tuple de sommets dans G en constitue une arête [3].

La "complexité en requêtes" d'un testeur est le nombre de requêtes qu'il effectue pendant son exécution.

On dira qu'une propriété est efficacement testable s'il existe un testeur ayant une complexité en requêtes indépendante de la taille de l'instance testée. C'est le cas par exemple pour le problème de la 3-coloration, comme nous le verrons plus loin.

3. Dans le cadre du modèle dense, qui est celui que nous utiliserons. Cf [12] pour plus de détails sur les autres requêtes possibles

Pour beaucoup de testeurs de propriétés, l'algorithme se décompose en deux parties. D'abord, le testeur effectue les requêtes nécessaires pour reconstituer le sous-graphe G' induit par un ensemble S (choisi aléatoirement) de sommets de G. Puis il exécute un algorithme exact sur ce sous-graphe, afin de déterminer s'il possède la propriété cherchée pour G. Enfin, il conclut que si G' n'a pas la propriété, G est ϵ-loin de l'avoir, et que si G' l'a, G l'a aussi. Si la taille de l'échantillon S ne dépend pas de celle de G, la propriété est bien efficacement testée.

Remarques : On pourrait plus généralement se donner n'importe quels p et $p' > 1/2$, mais il est d'usage courant de choisir 2/3, ce qui permet de comparer plus facilement les différents algorithmes existants et d'éviter de travailler sur des inégalités strictes.

Pour les graphes, une distance très courante est la distance de Hamming,(ou distance d'édition). Pour cette distance, un hypergraphe G est à distance ϵ d'un hypergraphe H si $\frac{|E(G)\Delta E(H)|}{n^k} = \epsilon$. Cette distance n'est définie que si G et H ont le même nombre n de sommets identifiés. Ceci peut sembler gênant, mais nous verrons qu'il n'en est rien, grâce à la propriété 1.

Pour les grands graphes, il est difficile voire impossible d'intégrer dans la mémoire vive de l'ordinateur utilisé la totalité du graphe sur lequel on raisonne, ce qui empêche a priori de faire un échantillonnage aléatoire simple du graphe. Cependant, plusieurs techniques de marches aléatoires existent ([21] par exemple) afin d'optimiser ce processus (par exemple basées sur le couplage rapide d'une chaine de Markov). Dans la suite, nous ferons donc l'hypothèse que constituer l'échantillon aléatoire du graphe est un processus rapide, dont nous ne tenons pas compte dans l'évaluation de la complexité des algorithmes.

Exemple 1 *Imaginons que nous voulions tester la propriété $\mathcal{P} = \{G : G$ est 3-colorable$\}$, avec pour distance critique ϵ. Dans ce cas, G sera ϵ-loin d'être 3-colorable s'il faut changer plus de $\epsilon.n^2$ arêtes à G pour qu'il le devienne.*

Un testeur pour cette propriété pourrait avoir la forme suivante :

Testeur(G) : Choisir un sous-graphe G' de G de taille S(G,ϵ).
Vérifier de manière exacte si G' est 3-colorable.
Si c'est le cas, accepter G.
Sinon, rejeter G.

Pour que cet algorithme soit valide, il faut montrer que :

- *Lorsque G est 3-colorable, G' a au moins 2 chances sur 3 de l'être. C'est bien le cas, G' l'est même à coup sur.*

- *Lorsque G est ϵ-loin d'être 3-colorable, G' a au moins 2 chances sur 3 de ne pas l'être. Pour cela on voit qu'il faut choisir correctement S*[4].

4. Pour cet exemple précis, on peut même montrer que S peut ne dépendre que de ϵ, cf [5] par exemple.

Nous pouvons remarquer que lorsqu'un hypergraphe G se projette dans un hypergraphe H, alors tous les sous-hypergraphes de G s'y projettent aussi. C'est particulièrement intéressant, car des résultats existent déjà concernant ce genre de propriétés, dites "monotones".

Définition 4 (Monotonie et hérédité) *Une propriété $\mathcal{P}$ est dite monotone lorsque pour tout hypergraphe G dans $\mathcal{P}$, tous les sous-hypergraphes de G sont aussi dans $\mathcal{P}$.*

Elle est dite héréditaire lorsque tous les sous-hypergraphes induits de G sont aussi dans $\mathcal{P}$. Toute propriété monotone est donc héréditaire.

Pour toute famille $\mathcal{F}$, on note $Forb_{ind}(\mathcal{F})$ la famille des hypergraphes n'ayant aucun élément de $\mathcal{F}$ comme sous-hypergraphe induit. Cette propriété est héréditaire.

Propriété 1 *Soit $\mathcal{P}$ une propriété héréditaire, G un graphe (ou plus généralement, un hypergraphe) n'ayant pas cette propriété. Alors le graphe $G_0 \in \mathcal{P}$ le plus proche de G est un graphe partiel de G.*

Preuve Soit $\mathcal{P}$ une propriété héréditaire, G un graphe n'ayant pas cette propriété. Soit $G_0 \in \mathcal{P}$ le graphe le plus proche de G pour la distance de Hamming. Soit $E_0 = E(G)\Delta E(G_0)$. Par définition, $d(G, G_0)=|E_0|/n^2$. On peut écrire E_0 comme union d'arêtes de G et de G_0 : $E_0 = E_0^G \cup E_0^{G_0}$

On définit alors $G_1=G \cap G_0$. Alors, comme $\mathcal{P}$ est héréditaire, $G_1 \in \mathcal{P}$. Par ailleurs $|E(G)\Delta E(G_1)| = |E_0^G| \leq |E_0|$, donc $d(G, G_1) \leq d(G, G_0)$ avec égalité si et seulement si $|E_0^{G_0}| = 0$, i.e ssi $G_0=G_1$.

Par définition de G_0, on a donc bien $G_0 = G_1$, donc G_0 est un graphe partiel de G.

□

Cette propriété implique que lorsqu'on veut évaluer la distance d'un graphe à une propriété héréditaire, il suffit d'évaluer le nombre minimal n_{min} d'arêtes qu'il faut lui enlever pour qu'il aie la propriété. Ceci nous permet de définir précisément la distance d'un graphe à une propriété héréditaire via la distance de Hamming : ce sera n_{min}/n^2, où n est le nombre de sommets du graphe.

La propriété suivante est triviale, mais elle sera utile pour comprendre notre démarche plus tard.

Propriété 2 *Si une propriété $\mathcal{P}$ est héréditaire, alors il existe $\mathcal{F}$ telle que $\mathcal{P} = Forb_{ind}(\mathcal{F})$*

Preuve Choisir $\mathcal{F} = \{H : H \text{ tel que } H \notin \mathcal{P}\}$ convient. □

2.1 Cas de l'homomorphisme

Dans toute la suite de ce rapport, nous chercherons à répondre à la question : "Est-ce que le graphe (ou l'hypergraphe) G se projette dans H ?". G sera donc toujours le graphe source, et H le graphe cible. On notera couramment $V(G)$,

$E(G)$, $V(H)$, $E(H)$ les sommets et arêtes (ou hyperarêtes) respectivement de G et de H.

Tout d'abord rappelons ce qu'est un homomorphisme de graphes :

Définition 5 (Homomorphisme de graphes) *Soient G et H deux graphes. $f : V(G) \to V(H)$ est un homomorphisme de graphe si : $\forall (u,v) \in E(G), (f(u)f(v)) \in E(H)$*

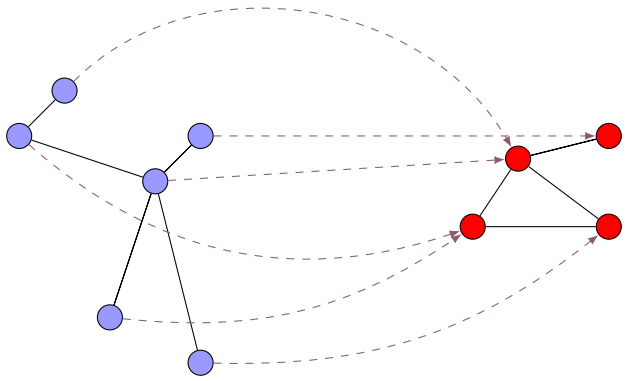

FIGURE 2 – Un exemple d'homomorphisme entre deux graphes

*Remarque :*Un graphe G est k-colorable si et seulement si G se projette dans K_k, où K_k est le graphe complet sur k sommets.

2.1.1 Quelle propriété à tester ?

Nous pouvons envisager initialement le problème de deux manières. La première manière de procéder serait de fixer H, et échantillonner G : nous écririons alors un testeur pour la propriété $Hom_H = \{G$: il existe un homomorphisme de G vers $H\}$.

La deuxième serait de fixer G, et d'échantillonner H, pour tester la propriété Im_G=$\{H$:il existe un homomorphisme de G vers $H\}$. Dans le cadre par exemple d'une requête à une base de données, cette situation semble plus naturelle, vues les tailles respectives des graphes représentant G et H.

2.1.2 Etude préliminaire

Notre première contribution est la démonstration suivante qui montre que cette approche ne présente que peu d'intérêt.

Théorème 1 *Im_G n'est pas efficacement testable.*

Preuve Pour montrer que Im_G n'est pas efficacement testable, il suffit de trouver un G pour lequel il existe un nombre infini de H ne pouvant être testés correctement en temps constant.

Prenons $G = K_3$ et $H_i = K_3 \cup J_i$ pour $i \in \mathbb{N}$ (où $J_i = K_{i,i}$, le graphe complet biparti entre deux ensembles de i sommets).

Alors, pour tout i, $G \to H_i$. Cependant, prenons un sous-graphe aléatoire H'_i induit par s sommets de H_i. Alors :

$P(H'_i \subseteq J_i) = (1-\frac{3}{2i+3})^s \underset{i\to\infty}{\to} 1$, donc il n'existe aucune constante s telle que la série des H_i soit testable pour Im_{K_3} en échantillonant un nombre constant s de sommets des H_i : en effet, tous les H_i sont bien dans Im_G, cependant, on a pas P(tout sous graphe de taille s d'un H_i quelconque est aussi dans Im_G)>1/2. Nous ne pouvons donc pas avoir de testeur efficace pour la propriété. □

De plus, si on pose $2i + 3 = n$ et $s = cn$,

$$(1 - 3/n)^s \geq e^{\frac{3cn}{3-n}}, et\ e^{\frac{3cn}{3-n}} \geq 1/2 \Leftrightarrow c \leq \frac{n-1\ln 2}{3n},$$

ce qui signifie que l'on ne peut pas non plus envisager d'échantillonner les H_i de manière sublinéaire, ou linéaire avec une constante faible.

En conséquence de quoi, il n'est pas envisageable de tester Im_G. Cela nous enlève un champ important d'étude, puisque dans le cas classique de l'interrogation de bases de connaissances, la taille de la requête est telle qu'il n'est pas possible de l'échantillonner.

Nous pourrions également envisager une autre approche consistant à tester une propriété d'un graphe "hybride", construit à partir de G et H. Par exemple, nous pouvons construire un graphe composé K à partir de G et H tel que H se projette dans G si et seulement si il existe un automorphisme non trivial de K dans lui-même. Cependant, nous n'avons pas obtenu de résultat intéressant selon cette perspective.

C'est pourquoi nous restreindrons désormais notre étude au test de Hom_H. En effet nous ne nous attendons pas à pouvoir ainsi échantillonner une requête classique, tester si une base de connaissances se projette dans une autre est un problème classique lors, par exemple, de la fusion de telles bases pour éviter les redondances.

Nous allons donc tout d'abord justifier que cette propriété est effectivement testable.

Pour tout graphe H, on remarque que Hom_H est monotone (et donc héréditaire), et est connue pour être décidable.

L'existence d'un testeur pour cette propriété est assurée par un résultat de Rödl and Schacht ([19]) :

Corollaire 1 *Toute propriété héréditaire et décidable d'hypergraphes est efficacement testable.*

En vertu de ce corollaire, nous nous intéresserons donc exclusivement aux testeurs effectuant un nombre de requêtes indépendant de la taille de la donnée en entrée.

3 Approche statistique

Le corollaire 1 s'appuie sur un résultat plus général([19], Théorème 6)

Théorème 2 *Pour toute famille (éventuellement infinie) $\mathcal{F}$ d'hypergraphes k-uniformes, et pour tout $\epsilon > 0$, il existe des constantes $c > 0$ et $C > 0$ et n_0 telles qu'on ait le résultat suivant :*

Soit $G^{(k)}$ un hypergraphe k-uniforme sur $n \geq n_0$ sommets.

Si pour tout l dans $[C]$ [5] et tout $F^{(k)} \in \mathcal{F}$ sur l sommets, $G^{(k)}$ contient au plus cn^l copies induites de $F^{(k)}$, alors $G^{(k)}$ n'est pas ϵ-loin de $Forb_{ind}(\mathcal{F})$

Ce théorème nous assure que si un hypergraphe G est ϵ-loin d'une propriété héréditaire $\mathcal{P}$ alors il existe une taille critique n_0, et des constantes c et C positives telles que si $|V(G)| > n_0$, alors G induit au moins $cn^{|V(F^{(k)})|}$ copies d'un hypergraphe interdit $F^{(k)}$ de taille au plus C, donc a fortiori au moins cn^C copies d'un hypergraphe interdit $F'^{(k)}$ de taille C (en effet, nous avons vu que toute propriété héréditaire pouvait s'écrire sous la forme $Forb_{ind}(\mathcal{F})$).

Il nous suffit alors de trouver une constante S telle que tout sous-graphe induit de G de taille S aie une probabilité $\alpha > 1/2$ de contenir une de ces copies, et donc de ne pas être dans Hom_H. Un testeur de propriété simple pour Hom_H est alors :

3.1 Premier algorithme

> Si G est de taille inférieure à n_0, vérifier de manière exacte la propriété.
> Sinon, prendre aléatoirement un sous-graphe G' de G de taille S, vérifier de manière exacte s'il est dans Hom_H. Renvoyer pour G la réponse obtenue pour G'.

Si S ne dépend pas de n, cet algorithme a une complexité en requêtes, et donc un temps d'exécution, borné.

Théorème 3 $S = \sqrt[C]{\frac{\ln(2)}{c}}$ *convient.*

Preuve Pour tout sous-graphe induit G' de G de taille C, on a $P(G' = F'^{(k)}) = \frac{cN^C}{\binom{C}{N}}$

Alors la probabilité qu'un graphe induit G'' de G de taille S ne contienne pas $F'^{(k)}$ est $(1 - \frac{cN^C}{\binom{N}{C}})^{\binom{S}{C}})$, et on veut choisir S de façon à ce que cette probabilité soit strictement inférieure à $\frac{1}{2}$

Or

$$(1 - \frac{cN^C}{\binom{N}{C}})^{\binom{S}{C}}) \leq e^{-\frac{cN^C}{\binom{N}{C}} \cdot \binom{S}{C}} \leq e^{-c\binom{S}{C}} \leq e^{-cS^C}$$

5. i.e dans l'ensemble $1, ..., C$.

et

$$e^{-cS^C} < \frac{1}{2} \Leftrightarrow S > \sqrt[C]{\frac{\ln(2)}{c}}$$

□

3.2 Evalution de l'algorithme

Nous avons besoin pour exploiter ce résultat de pouvoir évaluer c et C.

Or ces constantes découlent de l'utilisation du lemme de régularité de Szeremedi ou d'une de ses variantes([4], [3], [6], [7] par exemple), lequel donne l'existence d'une constante qui prend des valeurs non utilisables en pratique.

Plus exactement, la valeur exacte de ces constantes est généralement inconnue, mais Gowers([16]) en a donné une borne inférieure (une tour de 2 de taille proportionnelle à $\log(\frac{1}{\epsilon})$) qui écarte l'espoir d'une application pratique : par exemple, si on prend le graphe associé au réseau facebook (où les arêtes représentent la relation "'être ami"), ce graphe a une densité d'environ 3×10^{-5}. On a donc une complexité bornée inférieurement par une tour de 2 de taille 10 si l'on veut échantillonner ainsi ce graphe, ce qui n'est pas utilisable en pratique.

4 Une autre manière d'échantillonner - transition par les CSP

En 2002, Andersson et Engerbretsen ont proposé un algorithme pour approximer le problème k-MAX-CSP [8]. En 2003, Alon et al. [2] en ont proposé un deuxième, et en 2010, Coja-Oghlan et al. en ont proposé un troisième, applicables à d'autres instances du problème [14].

Nous allons transformer le problème de l'homomorphisme de k-hypergraphes vers le problème de la résolution d'un système de k-CSP, et voir comment approximer k-max-CSP nous permet d'approximer également le problème de l'homomorphisme. Nous verrons aussi pourquoi approximer ce problème équivaut à écrire un testeur de propriété pour Hom_H

Nous verrons quelles propriétés nous pouvons donner à nos hypergraphes (étiquetés, orientés, uniformes ou non) tout en gardant cette équivalence, et leurs impacts sur la complexité de l'algorithme proposé.

4.1 Approximation du problème de l'homomorphisme

Nous donnons ici une traduction en terme de théorie des graphes de l'algorithme proposé dans [8]. Tout comme celui de Alon et al., il s'applique pour des instances denses de graphes (nous verrons dans la partie 4.4 ce qu'il faut comprendre par là), et ce dernier a une meilleure complexité asymptotique en requêtes. Cependant, lorsque l'on considère la complexité respective des algorithmes, celui de Andersson et Engerbretsen est utilisable sur des graphes de taille plus proche de celles qui nous intéressent.

Paramètres : ϵ, δ deux réels strictement positifs, $G(V,E)$, $H(W,F)$ deux hypergraphes k-uniformes, avec $|V| = n$ et $|W| = N$. Ces hypergraphes sont considérés orientés et non étiquetés.

Sortie : Un réel, p, tel que si on note $\hat{p} = \frac{1}{n^k} * |E_{max}|$, on aie $P(|\hat{p} - p| \leq 2\epsilon) \leq (1 - 2\delta)$, où E_{max} est le nombre d'arêtes du plus grand sous-graphe de G qui se projette dans H.

Nous avons donc "probablement" une approximation à $2\epsilon n^k$ près de cette taille maximale.

Remarque : pour l'instant, nous ne retournons que cette valeur, non pas la projection partielle elle-même.

Algorithme 1 : On pose : $l = \frac{2}{\epsilon}$, $t = \frac{32}{\epsilon^2} \ln \frac{8lN}{\epsilon\delta}$, $s = \frac{64k \ln N}{\epsilon^5} \ln \frac{16 \ln N}{\epsilon^2 \delta} + \frac{1}{\epsilon^2} \ln \frac{2}{\delta}$

1. Partitionner V en l sous-ensembles de taille n/l, V_1,...V_l.
2. Pour tout i, sélectionner aléatoirement (selon une distribution uniforme) dans $V \setminus V_i$ t $(k-1)$-tuples de sommets, on nomme ces ensembles U_i. On note $U = \bigcup_i U_i$
3. Générer aléatoirement (toujours uniformément) s k-tuples de sommets de G, on appelle cet ensemble C. On note en plus $C_i = C \cap V_i$, pour tout $i \in [l]$
4. Pour toute fonction $p : U \to W$, créer une fonction $\Pi_p^C : C \to W$ de la manière suivante :
   Pour tout $i \in [l]$, choisir une image pour les variables de C_i comme ceci :
   - Pour tout $v \in C_i$, trouver $w \in W$ qui maximise le nombre d'arêtes de $U_i \cup \{v\}$ qui sont correctement projetées par $p' : u \leftarrow p(u)$ si $u \in U_i$,$v \leftarrow w$.
   - On définit $\Pi_p^C(v) = w$
5. Calculer $S(p)$, le nombre d'arêtes de G formées par les sommets de C qui sont correctement projetées par Π_p^C.
6. Retourner $\frac{1}{s} S(\Pi_p^C)$ pour Π_p^C qui maximise cette valeur.

On obtient une complexité de $\exp(O(\frac{k \ln |W|}{\epsilon^3} \ln \frac{k|W|}{\epsilon\delta}))$ pour approcher $\hat{p}$ à 2ϵ près (un temps constant par rapport à la taille de G), et un temps linéaire en $|G|$ pour trouver l'assignation des sommets de G qui réalise cette approximation (en faisant l'hypothèse que tous les calculs, notamment ceux de l'étape 4, sont faits par des sous-algorithmes gloutons d'énumération complète des cas).

Pour trouver cette assignation, on utilise l'algorithme suivant :

Algorithme 2 :

1. Trouver la fonction p selectionnée à l'étape 6 de l'algorithme 1. On construit Π_p^V ainsi :
2. Pour tout i, et tout sommet v_i de V_i, trouver w qui maximise le nombre d'arêtes correctement projetées par par p' : $u \leftarrow p(u)$ si $u \in U_i$,$v \leftarrow w$. (#Le travail est déjà fait par l'algorithme 1 pour les sommets de C_i.)
3. Pour tout v, on choisit $\Pi_p^V(v) = w$

Pour montrer la validité de ces algorithmes, nous allons montrer comment le problème de l'homomorphisme d'hypergraphes se traduit en un autre, le problème k-MAX-CSP. La validité de ces algorithmes sera ensuite assurée par la validité de ceux de [8].

4.2 Problème k-MAX-CSP

Tout d'abord, pour définir un problème CSP, nous avons besoin des définitions suivantes [6] :

Définition 6 (Fonction de contrainte k-aire) *Soit D un ensemble appelé le domaine. Une fonction de contrainte k-aire sur D est une fonction de $D^k \rightarrow \{0,1\}$.*
Une famille de contraintes k-aires sur D est un ensemble de fonctions de contraintes k-aires sur D.

Par exemple, une fonction de contrainte 2-aire sur $[n]^2$ pourrait être f : $(a,b) \rightarrow a+b$ modulo 2
Une manière de simple de se représenter une fonction de contrainte est de la voir comme une fonction qui "accepte" une partie de D^k et "rejette" l'autre.

Définition 7 (Contrainte) *Soient $X = \{x_1, ...x_n\}$ un ensemble de variables, avec $n \geq k$, D un domaine, $\mathcal{F}$ une famille de contraintes (k-aires) sur D.*

Une contrainte sur X est un $(k+1)-tuple$ $(f, x_{i_1}, ..., x_{i_k}) \in \mathcal{F} \times X^k$. On dit que f dépend des variables $x_{i_1}, ..., x_{i_k}$, et on impose en plus que celles-ci soient toutes différentes.

Si aux variables d'une contrainte sont assignées les valeurs $(a_1, ..., a_k) \in D^k$, la contrainte prend la valeur $f(a_1, ..., a_k)$. Elle est dite satisfaite si cette valeur est 1.

Définition 8 (Problème max-$\mathcal{F}$) *Soient E, D et $\mathcal{F}$ définis comme précédemment. Une instance I du problème max-$\mathcal{F}$ est un ensemble de contraintes. Le problème max-$\mathcal{F}$ consiste à trouver une assignation des variables de E dans D qui satisfasse le maximum possible de contraintes de I.*

6. Ces définitions ne représentent pas le cas le plus général des problèmes CSP, mais sont ce dont nous avons besoin pour notre équivalence !

On propose la transformation classique suivante (voir par exemple [18]) :

Soient $G(V, E)$ et $H(W, F)$ deux hypergraphes k-uniformes. On se pose la question de savoir s'il existe un homomorphisme de G vers H. Plus précisément nous cherchons à assigner à chaque sommet de G un sommet de H de sorte que l'image de toute hyperarête de G soit une hyperarête de H.

Nous posons donc $X = V$ et $D = W$. Et nous définissons $f : (w_1, ..., w_k) \in D^k \rightarrow 1$ si $(w_1, ..., w_k) \in F$, 0 sinon.

Intuitivement, f "accepte" exactement les hyperarêtes de H. Nous définissons alors $\mathcal{F} = \{f\}$ [7]

Nous pouvons alors construire $I = \{(f, v_1, ..., v_k), (v_1, ..., v_k) \in E\}$.

Propriété 3 (Equivalence des problèmes) *Lorsque par une assignation des variables de X dans D on arrive à satisfaire toutes les contraintes de I, on a bien réussi à faire en sorte que toute hyperarête de G (i.e les éléments de E) soit une hyperarête de H (les éléments de F, exactement les tuples acceptés par f).*

Donc le problème de l'homomorphisme de G dans H est bien équivalent à l'instance du problème k-CSP définie selon cette transformation.

Avec cette transformation, résoudre le problème max-$\mathcal{F}$ pour I revient strictement à trouver la fonction de V dans D^k qui maximise le nombre d'hyperarêtes de G correctement projetées. Ou encore, cela revient à trouver le plus grand sous-hypergraphe de G (au sens du nombre d'arêtes) qui se projette dans H.

Nous voyons déjà une première restriction : toutes nos contraintes ont la même arité : k. Ceci implique de choisir G et H k-uniformes. De plus, une contrainte dépend de k variables différentes, ce qui signifie qu'un sommet ne peut intervenir plusieurs fois dans une hyperarête.

Par ailleurs, notre algorithme étant la traduction selon cette transformation de l'algorithme 2 de [8], il est correct.

4.3 Généralisations

Dans le cadre qui nous intéresse, les arêtes de G correspondent à des prédicats logiques. Ce qui signifie qu'elles sont typées : elles sont étiquetées, et on a une relation d'ordre partielle (une relation de "généralisation") sur ces étiquettes.

Concrètement, cela signifie que l'image v dans H d'une arête u de G doit avoir le même type que u, ou un type le généralisant.

Pour prendre en compte cette contrainte supplémentaire, nous allons modifier notre définition de $\mathcal{F}$. De plus, nous avons besoin d'une hypothèse supplémentaire : que l'ensemble des prédicats nous intéressant, et donc l'ensemble des étiquettes des hyperarêtes de nos hypergraphes, soit fini.

7. Ce concept de famille de contraintes semble artificiel. Cependant, il sera utile quand nous généraliserons cette transformation, cf partie 4.3.

Définition 9 (Famille des contraintes typées) *Soit (T,≤) l'ensemble des étiquettes, que nous appellerons des types possibles sur les hyperarêtes de nos hypergraphes, muni d'une relation d'ordre partielle : on écrit que $t_1 \leq t_2$ lorsque t_2 généralise t_1. Soit également H un hypergraphe.*

De plus, on note e la fonction qui à une hyperarête d'un hypergraphe quelconque associe son type.

Alors on peut remplacer $\mathcal{F}$ dans la propriété 3 ainsi : $\mathcal{F} = T$, et :

$\forall t \in T,\ \forall (w_1, ..., w_k) \in W^k,\ t(w_1, ..., w_k) = 1 \leftrightarrow ((w_1, ..., w_k) \in F$ *et* $e(w_1, ..., w_k) \leq t)$

Intuitivement, t accepte un tuple de W^k si ce tuple est une arête de H dont le type généralise t.

Définition 10 *Soit $\mathcal{F}$ une famille de contraintes sur W. Le nombre maximum de contraintes satisfiables sur un tuple pour $\mathcal{F}$, noté $\Sigma(\mathcal{F})$ est défini par :*

$$\Sigma(\mathcal{F}) = \max_{(w_1,...,w_k) \in W^k} |\{f \in \mathcal{F} : f(w_1, ..., w_k) = 1\}|$$

En terme d'hypergraphes, cette valeur correspond au plus grand nombre d'hyperarêtes affectant un même tuple de sommets. Les algorithmes de [8], s'appliquent encore, mais leur complexité dépend maintenant de $\Sigma(\mathcal{F})$. On obtient une complexité $\exp(O(\frac{\Sigma^3 k \ln |W|}{\epsilon^3} \ln \frac{\Sigma k |W|}{\epsilon \delta}))$.

Plus concrètement, cela signifie que si G et H sont des hypergraphes 3-uniformes de grande taille (plusieurs millions de sommets chacun), la complexité d'un algorithme exact sera a priori de l'ordre de $|H|^{|G|}$, et celle du testeur de propriété pour un facteur de distance de $0,1$ sera de l'ordre $|H|^{6*10^5}$, ce qui, bien qu'étant effectivement bien plus faible, reste toujours impraticable.

4.4 Remarques et limites

A priori, cet algorithme présente un avantage indéniable : il a une complexité qui ne dépend pas de de la taille de G, et qui ne croit que logarithmiquement avec la taille de H. Nous allons donner une explication, intuitive, d'une des raisons de ce succès. Ce qui expliquera aussi à quel moment intervient l'hypothèse que nous avions fait initialement de travailler sur des instances "denses" de CSP, et que nous n'avons jamais explicitement utilisée.

Si on note c la densité de G, pour que notre algorithme soit utile, il faut choisir $\epsilon \leq c$. Nous avons donc $|E| = cn^k$, et pour pouvoir choisir ϵ constant, il faut supposer que nous avons un d tel que pour tous les hypergraphes G que nous traiterons, $c \geq d$. Ceci implique que si l'on double la taille de G, pour garder la même densité, on doit avoir $|E|_{nouveau} = 2^k |E|_{ancien}$. La croissance du nombre d'arêtes est considérablement plus rapide que celle du nombre de sommets.

Prenons un exemple concret : un réseau social type Facebook, avec encore le graphe représentant le lien "être ami" entre les membres du réseau. Pour garder une densité constante, il faudrait que lorsque le nombre de membres double,

le nombre moyen de liens par membre quadruple. Ce qui ne correspond pas à ce que l'on observe empiriquement : le nombre moyen de liens tend à rester constant, ou à croître faiblement [13]. Dans ce genre de cas, la densité diminue, entraînant une diminution de ϵ, et une augmentation sensible de la complexité de l'algorithme.

Intuitivement, l'hypothèse du graphe dense est donc d'autant plus "forte" que le graphe est grand. Et elle explique le succès de ces algorithme sur les grands graphes, et on comprend mieux ce qui peut faire que cet algorithme soit en temps constant à partir d'une taille critique de G.

5 Bornes supérieures et inférieures pour la taille de l'échantillonnage nécessaire

5.1 L'importance de la cliquewidth de H

Nous allons maintenant donner une explication plus quantitative quand à l'importance de la structure de H, et notamment de la cliquewidth [8] de H.

Considérons p et n deux entiers, et, pour simplifier les notations, supposons que p divise n. Considérons les graphes complets K_p et K_n. A quelle distance K_n est-il de se projeter dans K_p ? Cette question est en fait équivalente à la suivante : combien d'arêtes au minimum faut-il enlever à K_n pour le rendre p-colorable ?

Propriété 4 *K_n est $\frac{n^2-n}{2pn^2}$-loin de se projeter dans K_p.*

Preuve Soit $\sigma : V(K_n) \to [p]$ une p-coloration (impropre) de K_n. Soit V_i la classe de coloration de la couleur i dans K_n. Alors, il faut supprimer toutes les arêtes (x, y) de K_n telles que $(x, y) \in V_i^2$ pour rendre cette p-coloration propre. Il faut donc supprimer $S = \frac{1}{2}\sum_{i=1}^{p} |V_i||V_i| - 1$ arêtes de V_n.

$f : \mathcal{R} \to \mathcal{R} : x \to x^2$ étant convexe, sous la contrainte $\sum_{i=1}^{n} |V_i| = n$, S est minimisée pour $|V_i| = \frac{n}{p}$, pour tout i.

Il faut donc supprimer au minimum $\frac{1}{2} * \sum_{i=1}^{p} (n/p)(n-1)/p$ arêtes à K_n pour le rendre p-colorable. □

Ce résultat amène deux corollaires concernant la p-coloration :

Corollaire 2 *Soit G un graphe tel $|V(G)| = n$, et $|E(G)| = \delta n^2$. Alors : $\forall p, 1 - \delta \leq \frac{n^2-n}{2pn^2} \Leftrightarrow G$ n'est pas p-colorable*

Preuve K_n est $1 - \delta$-loin de G, et $\frac{n^2-n}{2pn^2}$-loin d'être p-colorable. Donc si $1 - \delta \leq \frac{n^2-n}{2pn^2}$, G n'est pas p-colorable. □

8. i.e la taille de la plus grande clique.

Corollaire 3 *Aucun graphe n'est à distance supérieure à $\frac{1}{2p}$ d'être p-colorable.*

Ceci implique notamment qu'un testeur cohérent pour la p-coloration, et a fortiori pour l'homomorphisme vers tout graphe contenant une clique de taille p, doit choisir un paramètre de distance ϵ inférieur à $\frac{1}{2p}$. En conséquence, si notre base de données (le graphe H) contient de grandes cliques, quelque soit la densité de notre graphe d'entrée (G), il faut choisir ϵ très petit, ce qui amène à une explosion de la complexité des testeurs de propriétés.

5.2 Complexité d'un testeur canonique

Dans [1], Alon présente une construction permettant de construire, pour tout paramètre de distance ϵ, et tout graphe H un graphe qui soit ϵ-loin de ne pas contenir de copie de H. Nous allons présenter cette construction dans le cas où H est un triangle, et montrer en quoi elle interdit d'espérer faire un testeur générique pour l'homomorphisme (i.e. qui ne dépende pas du graphe H) qui aie une complexité en requêtes polynomiale en $1/\epsilon$. Cette construction est elle-même basée sur les graphes de Behrend ([11]).

Lemme 1 *Pour tout m suffisamment grand, il existe $X \subseteq [m]$ tel que $|X| = m^{1-g(m)}$, avec $g(m) = o(1)$, et tel que X ne contienne pas trois termes en progression arithmétique.*

Plus précisément, on peut montrer que ce résultat est valide pour $g(m) = c/log(m)$, où c est une petite constante. On peut alors définir un graphe de Behrend pour m et X, que nous noterons $B_{m,X}$.

Définition 11 (Graphe de Behrend $B_{m,X}$) *Soit m et X comme définis ci-dessus. Le graphe $B_{m,X}$ est alors le graphe suivant : il contient $6m$ sommets, séparés dans un ensemble V_1 de m sommets numérotés arbitrairement de 1 à m, V_2 de $2m$ sommets numérotés de 1 à $2m$, et V_3 de $3m$ sommets numérotés de 1 à $3m$.*

Alors, soit i, j, k des sommets de V_1, V_2 et V_3. Les arêtes de $B_{m,X}$ sont les suivantes : $(i,j) \in E(B_{m,X}) \Leftrightarrow j - i \in X$, $(i,k) \in E(B_{m,X}) \Leftrightarrow (k-i)/2 \in X$, $(j,k) \in E(B_{m,X}) \Leftrightarrow k - j \in X$.

Ce graphe a les propriétés immédiates suivantes :
- Il contient $3m|X|$ sommets, et comme $|X| = o(m)$, il n'est pas "dense", dans le sens qui nous intéresse.
- Ces sommets forment $m|X|$ triangles, chaque arête contribuant à un unique triangle. Il faut donc lui enlever $m|X|$ arêtes pour qu'il ne contienne plus aucun triangle.

Afin de donner des bornes intéressantes en terme de test de propriété, nous proposons une construction qui permet de "densifier" ce graphe tout en gardant des propriétés intéressantes.

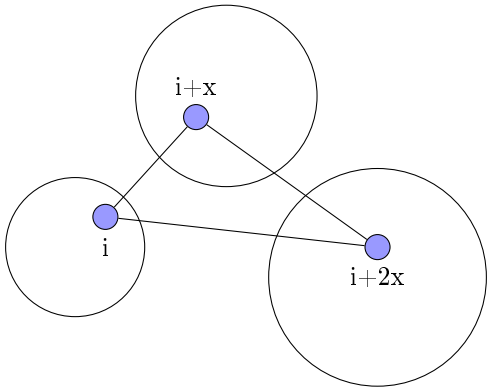

FIGURE 3 – Construction du graphe de Behrend, $x \in X$, $i \in [m]$

Définition 12 (s-blow up d'un graphe) *Soit G un graphe, $V(G) = (v_1, ..., v_n)$. On définit le s-blow-up de G de la manière suivante : on remplace tous les sommets v_i de G par des ensembles V_i de s sommets indépendants. Pour tout i et j dans $[n]$, si $(v_i, v_j) \in E(G)$, alors on forme un graphe bipartite complet entre V_i et V_j, sinon, on ne met aucune arête entre V_i et V_j*

Propriété 5 *Soient m,X et s. Alors le s-blow-up de $B_{m,X}$ a les propriétés suivantes :*

- *Il contient $6sm$ sommets et $3s^2m|X|$ arêtes*
- *Ces arêtes forment $|X|ms^3$ triangles, il faut lui en enlever $|X|ms^2$ pour tous les détruire.*

Preuve Chaque triplet de clusters formant un graphe tripartite complet génère s^3 triangles. Et ces triplets sont au nombres de m (clusters dans V_1) *|X| (clusters admissibles alors dans V_2) * 1 (l'unique cluster connecté aux deux premiers choisis). On a donc bien $|X|ms^3$ triangles.

Chaque arête participe à exactement s triangles : si l'arête relie, par exemple, un sommet u d'un cluster de V_1 et un sommet v d'un cluster de V_2, alors il y a exactement s sommets de V_3 réunis dans un unique cluster qui forment des triangles contenant l'arête (uv). Il faut donc enlever au minimum $|X|ms^2$ arêtes pour déconnecter tous ces triangles.

A l'inverse, si on enlève toutes les arêtes reliant des sommets de V_1 et V_2, on en a enlevé strictement $|X|ms^2$ et on est sur d'avoir bien déconnecté tous les triangles. □

On notera ce s-blow-up $B_{m,X,s}$.

Il nous faut maintenant un graphe duquel $B_{m,X,s}$ soit loin de se projeter. On définit, pour m, X, s fixés, $H_{m,X,s}$ comme étant l'union disjointe de tous les sous graphes de $B_{m,X,s}$ ne contenant pas de triangles.

Soit $\epsilon > 0$. On choisit m tel que $m^{-g(m)} > 36\epsilon$, et $X \subseteq [m]$ tel que $|X| \geq m^{1-g(m)}$. Par exemple, si on écrit $g(m) = c/\sqrt{\log m}$, $m = \frac{36}{\epsilon}^{\frac{1}{c^2}\log\frac{36}{\epsilon}}$ convient. On remarque alors que m peut dont être choisi superpolynomial [9] en $1/\epsilon$.

On se donne alors $n \gg m$. On pose $s = n/6m$. Il faut alors enlever au moins $|X|.m.s^2$ arêtes à $B_{m,X,s}$ pour qu'il ne contienne aucun triangle. Or

$$|X|.m.s^2 \geq m^{1-g(m)}.m.n^2/36m^2 \geq m^{-g(m)}n^2 \geq \epsilon n^2$$

.

Donc $B_{m,X,s}$ est au moins ϵ-loin de se projeter dans $H_{m,X,s}$.

Soit maintenant $q \leq n$, et H' un sous graphe de $B_{m,X,s}$ induit par q sommets. Alors la probabilité que H' contienne un triangle est au maximum

$$\binom{q}{3}\frac{|X|ms^3}{n^3} \leq q^3\frac{m^{2-g(m)}s^3}{216m^3s^3} \leq \frac{q^3m^{-g(m)}}{216m}$$

Alors pour que cette probabilité soit plus grande que $1/2$ pour tout m, on doit avoir $q = \Omega(m^{1/3})$, donc q superpolynomial en $1/\epsilon$.

Nous pouvons tirer deux conséquences de ce résultat.

Premièrement, on ne peut pas borner la complexité en requêtes d'un testeur canonique (du type "On échantillonne un sous-graphe, on cherche un homomorphisme depuis ce sous-graphe") qui prendrait à la fois G et H en entrée pour l'homomorphisme par un polynôme en $1/\epsilon$. On ne peut donc construire de tels testeurs qu'à H fixé.

La deuxième conséquence est que l'homomorphisme n'est pas efficacement testable dans le cas non dense : si on s'autorise à prendre des graphes G non denses (et qu'on change notre mesure en conséquence), il existe une famille infinie de graphes pour laquelle la taille des échantillons nécessaires est $\Omega(n^1/3)$ pour tester l'homomorphisme dans les graphes de cette famille. Plus précisément, si on s'autorise à prendre des graphes G dont le nombre d'arêtes est $O(n^{2-\epsilon})$, avec ϵ arbitrairement petit, et qu'on change notre mesure en conséquence, l'homomorphisme n'est plus efficacement testable.

6 Conclusion

Le test de propriété pour l'homomorphisme de graphes se heurte à plusieurs difficultés.

Tout d'abord, nous avons montré qu'il n'est pas envisageable d'échantillonner de manière aléatoire notre base de connaissances (1) et conservant une taille d'échantillon sublinéaire en la taille de celle-ci, bien que ce soit typiquement la plus grosse de nos données et donc le cas qui nous intéresse le plus.

Ensuite, même dans le cadre de la fusion de bases de connaissances, nous avons besoin d'hypothèses fortes à la fois sur la densité de cette requête et sur l'absence de grosses cliques dans la base de données pour pouvoir espérer tester

9. i.e, si on écrit $m = f(1/\epsilon)$, pour toute fonction polynomiale g, $g = o(f)$

la requête. Et on ne peut pas relâcher cette hypothèse de densité, même très faiblement, sans tomber dans des complexités en requêtes superpolynomiales. Ces résultats concernants le test de la requête (que nous avons appellé le test de la propriété Hom_H) sont résumés dans la 4.

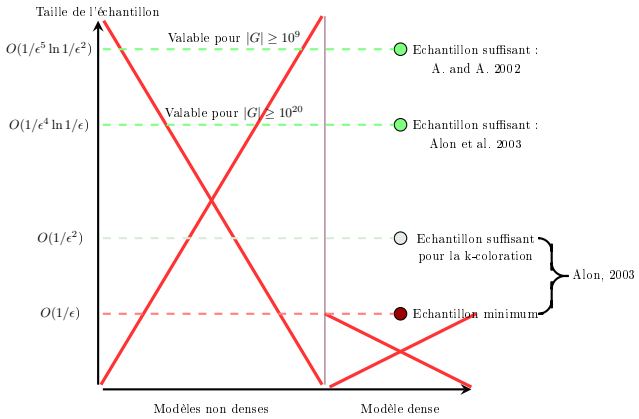

FIGURE 4 – Résultats concernant le test de Hom_H

En conséquence, il semble plus intéressant de se tourner vers des techniques semi-déterministes, par exemple pour indexer la base de données (décomposition en modèles fréquents, indexation des requêtes fréquentes,...).

Cependant, concernant le test de propriété sur les graphes, nous n'avons pas non plus étudié la possibilité d'étudier un graphe fonction de G et H. Une question ouverte est donc de savoir si un tel graphe est testable, et si oui, avec quelle volume d'échantillonnage.

Par ailleurs, nous nous sommes concentrés sur les testeurs efficaces (i.e dont la complexité en requêtes est indépendante de l'entrée) pour le test de Hom_H. Une autre question ouverte est de déterminer s'il est possible de tester Hom_H de manière simplement sublinéaire, et quelle taille typique d'échantillon il est alors possible d'obtenir.

Références

[1] Noga Alon. Testing subgraphs in large graphs. *Random Struct. Algorithms*, 21 :359–370, October 2002.

[2] Noga Alon, W. Fernandez de la Vega, Ravi Kannan, and Marek Karpinski. Random sampling and approximation of max-csps. *J. Comput. Syst. Sci.*, 67 :212–243, September 2003.

[3] Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy. Efficient testing of large graphs. *Combinatorica*, 20 :451–476, 2000.

[4] Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties : It's all about regularity. *SIAM Journal on Computing*, 39(1) :143–167, 2009.

[5] Noga Alon and Michael Krivelevich. Testing k-colorability. *SIAM J. Discret. Math.*, 15 :211–227, February 2002.

[6] Noga Alon and Asaf Shapira. A characterization of the (natural) graph properties testable with one-sided error. In *Proc. of FOCS 2005*, pages 429–438, 2005.

[7] Noga Alon and Asaf Shapira. Every monotone graph property is testable. *SIAM Journal on Computing*, 38(2) :505–522, 2008.

[8] Gunnar Andersson and Lars Engebretsen. Property testers for dense constraint satisfaction programs on finite domains. *Random Struct. Algorithms*, 21 :14–32, August 2002.

[9] Jean-françois Baget. *Représenter des connaissances et raissonner avec des hypergraphes : de la projection à la dérivation sous contraintes.* PhD thesis, Ecole Doctorale de Montpellier II, 2001.

[10] Jean-François Baget. Homomorphismes d'hypergraphes pour la subsomption en rdf/rdfs. *L'OBJET*, 10(2-3) :203–216, 2004.

[11] F Behrend. On the sets of integers which contain no three in arithmetic progression. In *Proceedings of the National Academy of Sciences 23*, pages 331–332, 1946.

[12] I. Ben-Eliezer, T. Kaufman, Michael Krivelevich, and Dana Ron. Comparing the strength of query types in property testing : The case of testing k-colorability. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1213–1222. Society for Industrial and Applied Mathematics, 2008.

[13] Jean-Hugues Bretin. website. http ://www.seomanager.fr/statistiques-facebook-fevrier-2010.html.

[14] Amin Coja-Oghlan, Colin Cooper, and Alan Frieze. An efficient sparse regularity concept. *SIAM Journal on Discrete Mathematics*, 23(4) :2000–2034, 2010.

[15] Oded Goldreich. Introduction to testing graph properties. Technical Report TR10-082, Electronic Colloquim on Computational Complexity, may 2010.

[16] W.T. Gowers. Lower bounds of tower type for szemerédi's uniformity lemma. *Geometric And Functional Analysis*, 7 :322–337, 1997.

[17] Pavol Hell and Jaroslav Nesetril. *Graphs and Homomorphisms.* Oxford Press, 2005.

[18] Michel Chein et Marie-Laure Mugnier. *Graph-Based Knowledge Representation : Computational Foundations of Conceptual Graphs.* Advanced Information and Knowledge Processing. Springer, 2008.

[19] Vojtech Rodl and Mathias Schacht. Generalizations of the removal lemma. *Combinatorica*, 29(4) :467–501, 2009.

[20] Dana Ron. Algorithmic and analysis techniques in property testing. *Found. Trends Theor. Comput. Sci.*, 5(2) :73–205, 2010.

[21] Satu Elisa Schaeffer. Scalable uniform graph sampling by local computation. *SIAM J. Scientific Computing*, 32(5) :2937–2963, 2010.

[22] John F. Sowa. Conceptual graphs for a data base interface. *IBM J. Res. Dev.*, 20 :336–357, July 1976.

[23] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2) :357–365, 2005.

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399