4 rue Léonard de Vinci
BP 6759
F-45067 Orléans Cedex 2
FRANCE
http://www.univ-orleans.fr/lifo

Rapport de Recherche

ANR AGAPE - Implémentation du problème de couverture minimum et applications d'algorithmes exacts paramétrés aux graphes aléatoires

Vincent Levorato
LIFO, Université d'Orléans

Rapport n^o **RR-2011-16**

Table des matières

1 Introduction **2**

2 Algorithmes exacts pour le problème du Minimum Vertex Cover **2**
2.1 Algorithme Minimum Vertex Cover - Brute Force 3
2.2 Algorithme Minimum Vertex Cover - DegMax et Kernelization . 4
2.3 Algorithme Minimum Vertex Cover - Degree Branching Strategy 6
2.4 Algorithme Minimum Vertex Cover - Niedermeier 7

3 Applications d'algorithmes à d'autres classes de graphes aléatoires et/ou réguliers **10**
3.1 Graphes aléatoires non réguliers 10
3.1.1 Modèle de Barabási-Albert 10
3.1.2 Modèle d'Eppstein 12
3.1.3 Modèles Small-World 14
3.2 Graphes non aléatoires 18
3.2.1 Graphe sous forme de grille 18
3.2.2 Graphe sous forme de treillis d'anneaux 18
3.3 Graphes aléatoires réguliers 19
3.4 Exploration du voisinage de graphe 21
3.5 Récapitulatif des résultats 22

4 Conclusion - Perspectives **23**

1 Introduction

Ce document est axé autour de deux parties principales : l'implémentation d'algorithmes pour le problème du *Minimum Vertex Cover* et l'exposé de résultats concernant le comportement d'algorithmes *Minimum Vertex Cover* et *Maximum Independent Set* sur des graphes aléatoires et/ou réguliers particuliers.

2 Algorithmes exacts pour le problème du Minimum Vertex Cover

Il existe de nombreux problèmes difficiles qui exigent un temps d'exécution exponentielle quand la complexité est mesurée en fonction de la taille de l'entrée, mais qui sont aussi calculables en temps polynomial en la taille de l'entrée et exponentielle pour un paramètre k. Les problèmes dans lesquels un certain paramètre k est fixé sont appelés *problèmes paramétrés*. Un tel problème est dit FPT (*fixed-parameter tractable* ou *résoluble à paramètre fixé*).
Dans les graphes, le problème du Minimum Vertex Cover appartient à la classe des problèmes FPT. Pour le résoudre, on a besoin d'approximer k, ce qui se fait habituellement par une 2-approximation. Dans un graphe donné, on construit

une couverture de sommets en prenant les deux sommets extrémités d'une arête en les enlevant du graphe et en recommençant jusqu'à ce que le graphe soit vide. Cet algorithme simple donne une couverture qui est au plus le double de la taille d'une couverture minimum. Dans ce travail, une approximation gloutonne qui donne une meilleure approximation, a été utilisée à la place. A chaque itération, l'algorithme glouton choisit le sommet de plus haut degré et l'ajoute à la couverture, jusqu'à ce que toutes les arêtes du graphe soient couvertes. Les algorithmes de Minimum Vertex Cover qui ont été étudiés sont :

- l'algorithme Brute Force dont la complexité est en $O(2^k n)$.
- un algorithme branchant à chaque étape sur le sommet de degré maximum et ses voisins avec une kernelization [BG93] avant chaque branchement.
- un algorithme de branchement simple en fonction des degrés des sommets du graphe ([Nie06] p. 90) dont la complexité est en $O(1.47^k n)$.
- l'algorithme de branchement de Niedermeier ([Nie06] p. 98-101) dont la complexité est en $O(1.33^k n)$.

A la fin de cette section sur la figure 5 se trouve la comparaison des temps d'exécution des algorithmes étudiés ici avec en plus la taille k de la couverture minimum.
Note : concernant l'implémentation du branchement, celle-ci a été modifiée de manière à ne plus faire de copie de graphe. Quand on supprime un sommet ou un ensemble de sommets d'un graphe G, ceux-ci sont gardés en mémoire avec leurs arêtes respectives dans un sous-graphe G'. Quand l'algorithme remonte de la branche, on fusionne G et G' ($G = G \cup G'$). Cela permet de ne pas surcharger le facteur polynomial de l'algorithme inutilement avec des copies de graphes.

2.1 Algorithme Minimum Vertex Cover - Brute Force

Cet algorithme parcourt les arêtes du graphe et branche selon les sommets situés à chaque extrémité de l'arête. Toute couverture contient au moins un des deux sommets qui composent l'arête. On obtient ainsi un algorithme récursif qui renvoie une couverture minimum du graphe. L'algorithme basique (noté `VC_BF(G,k)`) renvoyant vrai ou faux à la question « existe-t-il une couverture minimum de taille k ? » s'exécute ainsi :

1. Si $|E| = 0$ renvoyer VRAI
2. Si $|E| \geq k \times |V|$ retourner FAUX
3. Prendre une arête $e = xy$ de G,
   et retourner `VC_BF`$(G - \{x\}, k-1) \vee$ `VC_BF`$(G - \{y\}, k-1)$

Une légère modification de l'algorithme nous renvoie la couverture elle-même. L'algorithme est FPT, mais sa complexité étant en $O(2^k n)$, le temps d'exécution commence à poser problème quand k est supérieur à 30. La figure 1 montre les résultats obtenus sur des graphes aléatoires d'Erdös et Rényi (avec n le nombre de sommets et p la probabilité de connexité dont la densité découle). En toute logique, plus la densité du graphe augmente (autrement dit plus p tend vers

1), plus l'algorithme prend du temps à s'exécuter ($p = 1$ équivaut à un graphe complet).

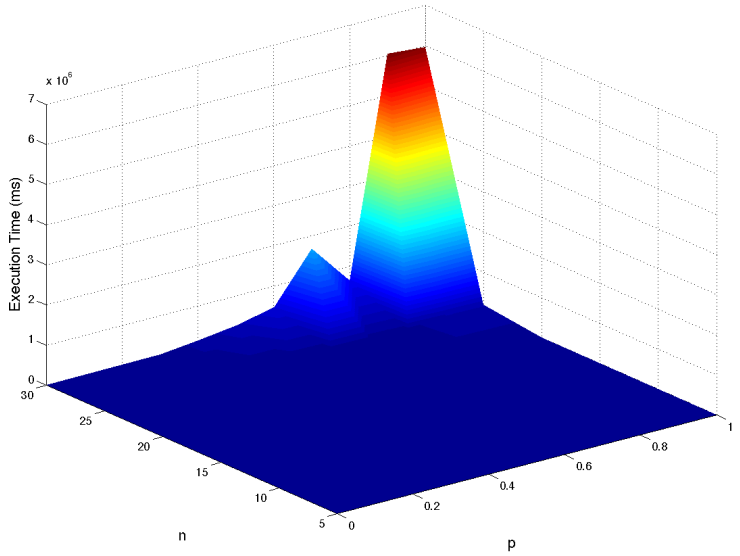


FIGURE 1 – Temps d'exécution de l'algorithme Minimum Vertex Cover - Brute Force

2.2 Algorithme Minimum Vertex Cover - DegMax et Kernelization

Le but d'une kernelization (ou nucléarisation en français) est d'obtenir une réduction d'un problème par une fonction calculable en temps polynomial. On réduit le graphe à un noyau "incompressible", graphe que l'on peut passer en paramètre de n'importe quel algorithme FPT de branchement par exemple. Dans notre cas, nous utilisons la kernelization connue de Buss et Goldsmith [BG93] qui est composée des règles de réduction suivantes :

1. Tant qu'il existe un sommet x de degré 0, enlever x de G.
2. Tant qu'il existe un sommet x de degré 1, enlever x de G et ajouter son voisin à la couverture.
3. Tant qu'il existe un sommet x de degré $> k$, enlever x de G et l'ajouter à la couverture.

En couplant cette kernelization à un algorithme de branchement sur le sommet de degré maximum et ses voisins, on obtient l'algorithme `VC_Kern` suivant :

1. Si $|E| = 0$ renvoyer VRAI

2. Si $|E| \geq k \times |V|$ retourner FAUX
3. $G' =$ BussKern(G)
4. Soit v_{max} le sommet de degré maximum de G',
   retourner `VC_Kern`$(G' - \{v_{max}\}, k-1) \vee$ `VC_Kern`$(G' - N(v_{max}), k - |N(v_{max})|)$

Ici encore, on ne présente que l'algorithme qui renvoie VRAI ou FAUX à la question existe-t-il une couverture minimum de taille k. Les versions implémentées renvoie un ensemble de sommets correspondant à une couverture minimum du graphe. Sur la figure 2 est représenté le temps d'exécution de l'algorithme sur des graphes d'Erdös-Rényi pour n de 5 à 100 par pas de 5, et p de 0.0 à 1.0 par pas de 0.1. On remarque, à l'instar des algorithmes pour le problème Maximum Independent Set, que le pire temps apparaît pour $p = 0.2$ (le pire p pour MIS variait entre 0.1 et 0.2 selon les méthodes).

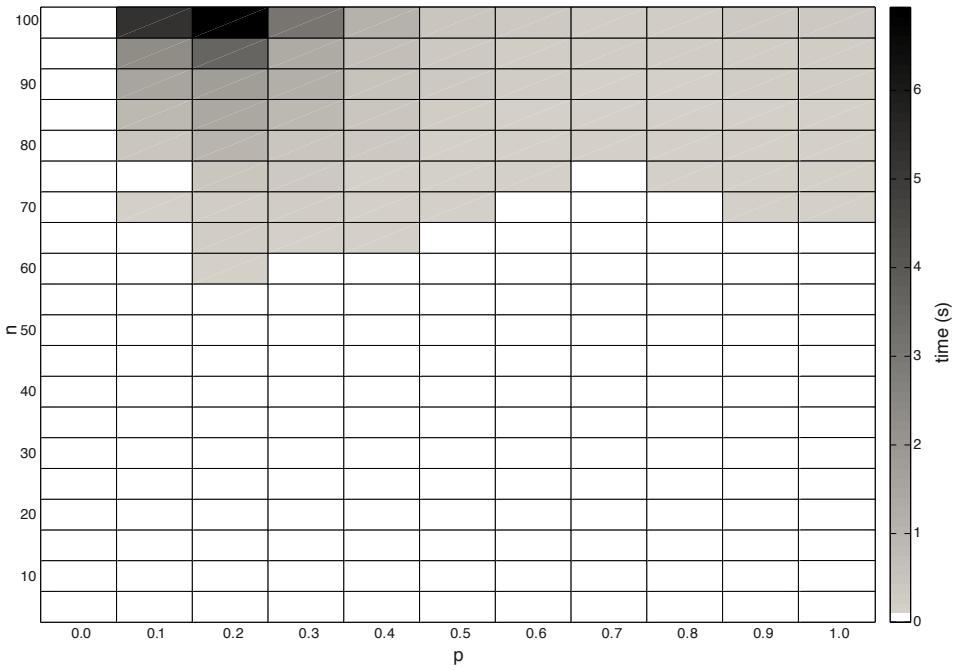


FIGURE 2 – Temps d'exécution de l'algorithme Minimum Vertex Cover - DegMax et Kernelization (BussG93)

2.3 Algorithme Minimum Vertex Cover - Degree Branching Strategy

Cet algorithme tiré de [Nie06] (p. 90) se composent de trois règles de branchement simples permettant de réduire de façon significative la taille de l'arbre des solutions de la méthode Brute Force. Cet algorithme que l'on notera `VC_DBS` suit les étapes suivantes :

1. Si $|E| = 0$ renvoyer VRAI
2. Si $|E| \geq k \times |V|$ retourner FAUX
3. S'il existe un sommet v_1 de degré 1 dans G,
   retourner $\texttt{VC\_DBS}(G - N(v_1), k - 1)$
4. S'il existe un sommet v_2 de degré 2 dans G,
   retourner $\texttt{VC\_DBS}(G - N^2[v_2], k - |N^2[v_2]|) \vee \texttt{VC\_DBS}(G - N(v_2), k - |N(v_2)|)$
5. S'il existe un sommet v_3 de degré 3 ou plus dans G,
   retourner $\texttt{VC\_DBS}(G - \{v_3\}, k - 1) \vee \texttt{VC\_DBS}(G - N(v_3), k - |N(v_3)|)$

Cet algorithme a une complexité en temps d'exécution de $O(1.47^k n)$. L'exécution de celui-ci sur des graphes d'Erdös-Rényi apparaît en figure 3.

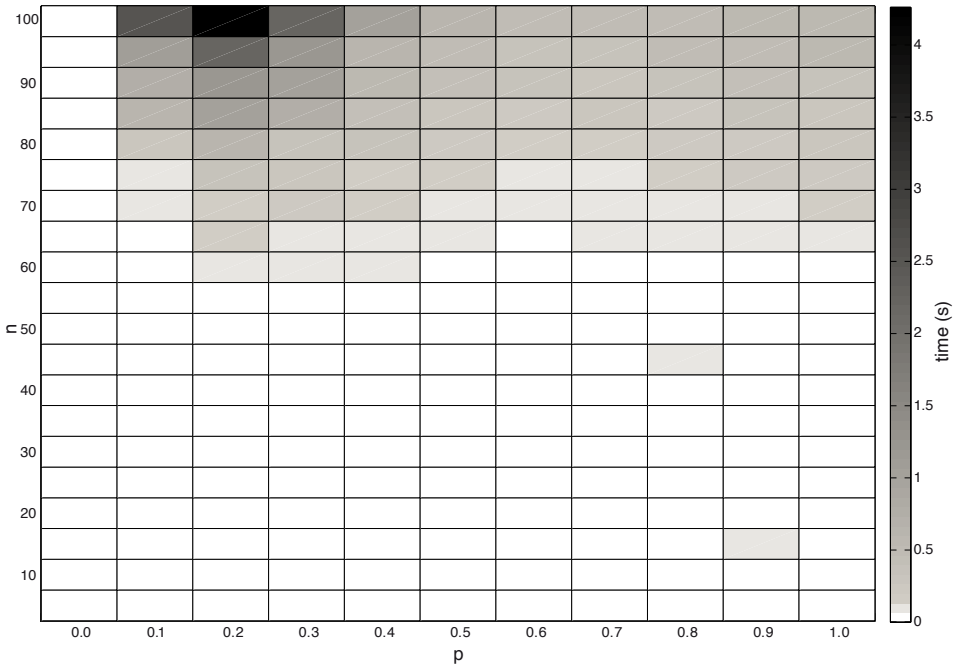


FIGURE 3 – Temps d'exécution de l'algorithme Minimum Vertex Cover - Degree Branching Strategy

2.4 Algorithme Minimum Vertex Cover - Niedermeier

Cet algorithme [Nie06] (p. 98-101) est également un algorithme de branchement utilisant des règles mais bien plus fin que le précédent, prenant en compte beaucoup plus de cas dont la complexité en temps d'exécution est $O(1.33^k n)$. La figure 4 visualise les résultats de l'algorithme sur les graphes d'Erdös-Rényi. Nous invitons le lecteur à consulter la référence ci-avant pour une description complète de l'algorithme qui suit les grandes étapes suivantes :

1. S'il existe un sommet x de degré 1, brancher selon $N(v)$.
2. S'il existe un sommet x de degré ≥ 5, brancher selon x et $N(x)$.
3. Si le graphe est régulier, prendre un sommet x au hasard et brancher selon x et $N(x)$.
4. S'il existe un sommet x de degré 2, avec a et b ses voisins, alors :
   (a) S'il existe une arête entre a et b, brancher selon $\{a, b\}$.
   (b) S'il n'existe pas d'arête entre a et b, que $deg(a) = deg(b) = 2$, et que $\exists c \in N(a) \cap N(b)$ avec $c \neq x$, brancher selon $\{x, c\}$.
   (c) Sinon brancher selon $N(x)$ et $N(a) \cup N(b)$.
5. S'il existe un sommet x de degré 3, ayant pour voisins a, b, c alors :
   (a) Si x forme un triangle avec de 2 de ses voisins (par exemple {x,a,b}), brancher selon $N(x)$ et $N(c)$.
   (b) Si x appartient à un cycle de longueur 4 constitué de $\{x, a, b, d\}$ où d est un autre sommet non voisin de x, brancher selon $N(x)$ et $\{x, d\}$.
   (c) Si $N(a) \cap N(b) \cap N(c) = \emptyset$ et que l'un des voisins de x a un degré égal à 4, par exemple $|N(a)| = 4$, alors brancher selon $N(x)$, $N(a)$, et $\{a\} \cup N(b) \cup N(c)$.

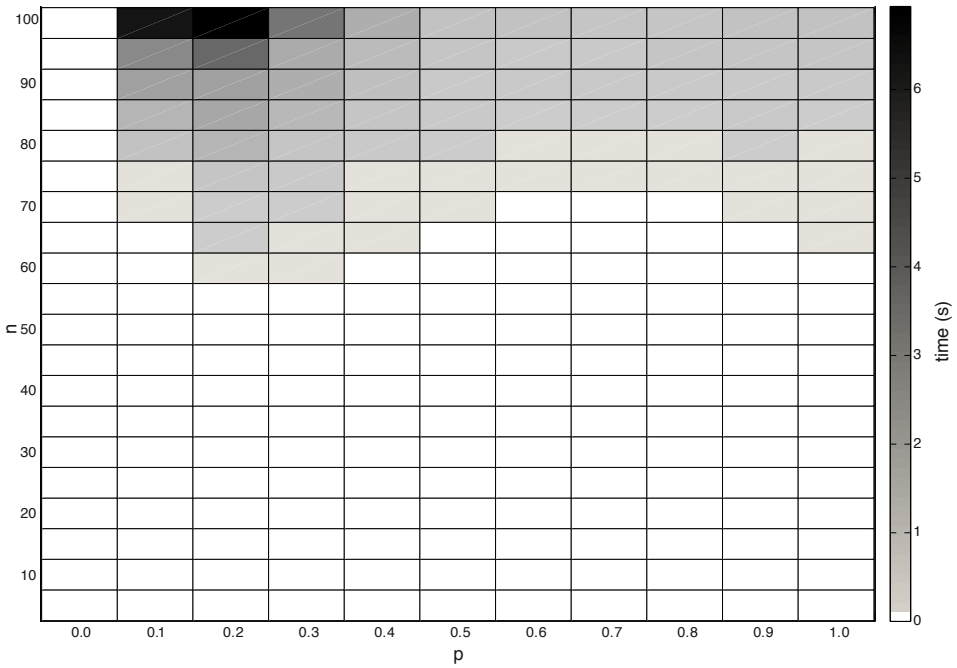


FIGURE 4 – Temps d'exécution de l'algorithme Minimum Vertex Cover - Niedermeier06

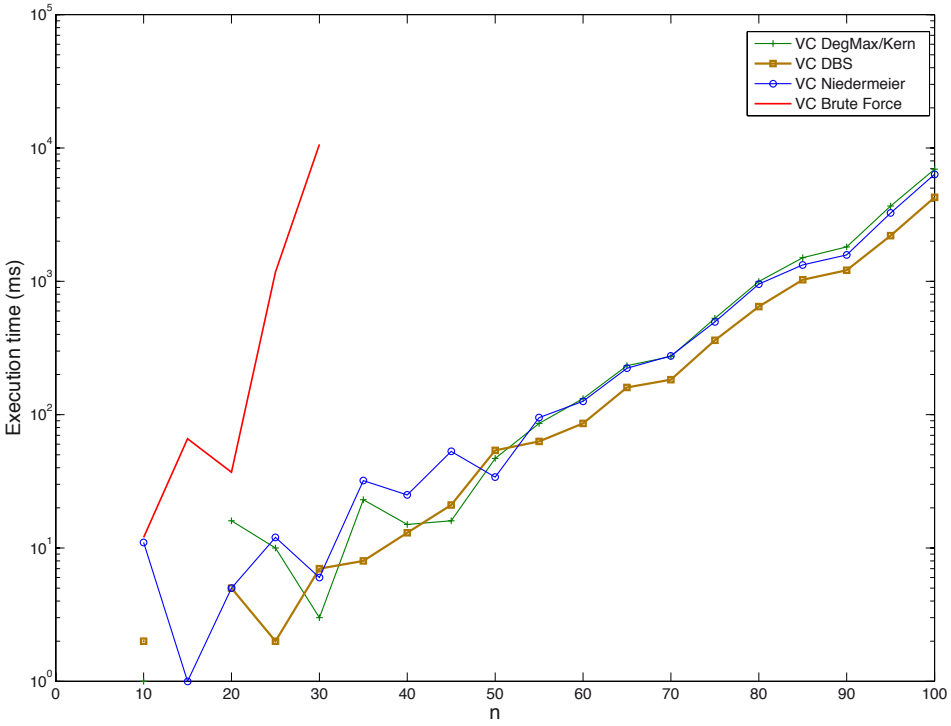


FIGURE 5 – Temps d'exécution des algorithmes Minimum Vertex Cover sur des graphes ER pour $p = 0.2$ (échelle log sur y)

3 Applications d'algorithmes à d'autres classes de graphes aléatoires et/ou réguliers

Nous avions émis des doutes sur l'utilisation de jeux de données basés sur les graphe aléatoires d'Erdös-Rényi restreignant ainsi fortement notre étude à un type de graphe particulier (les graphes aléatoires ne sont pas des graphes quelconques) possédant certaines propriétés dont les plus connues sont la distribution des degrés suivant une loi de Poisson et l'émergence d'une composante connexe géante au moment de la création du graphe. Afin de trouver certains types de graphes permettant de se rapprocher de la borne théorique des algorithmes étudiés, certains modèles sont ici proposés et testés avec l'algorithme Maximum Independent Set (noté MIS) de [FGK09] et les algorithmes Minimum Vertex Cover (noté MVC) décrits dans la précédente section que nous noterons DBS, DegMaxKern, et NiedermeierB. Les modèles de graphes étudiés apparaissent dans le tableau 1.

	Aléatoire	**Non Aléatoire**
Régulier	Random Regular Graphs	Ring Lattice graphs
Non Régulier	Barabási-Albert model Eppstein model Kleinberg small-world model Watts and Strogatz small-world model	Grid graphs

Table 1 – Modèles de graphes testés avec MIS et MVC

3.1 Graphes aléatoires non réguliers

3.1.1 Modèle de Barabási-Albert

Le modèle Barabási-Albert (BA) est un algorithme de génération aléatoire de graphe à invariant d'échelle utilisant un mécanisme d'attachement préférentiel [BA99]. Le graphe initial est composé de n_0 noeuds, les nouveaux nœuds étant ajoutés à ceux du graphe un par un. Chaque nouveau nœud est connecté à k nœuds existants avec une probabilité qui est proportionnelle au nombre d'arêtes auxquelles sont rattachés les noeuds, jusqu'à atteindre le nombre n de noeuds voulus. A un pas donné, la probabilité p de la création d'une arête entre un sommet v existant et le nouveau sommet est $p = \frac{deg(v)+1}{|E|+|V|}$ où $|E|$ et $|V|$ sont respectivement le nombre d'arêtes et de sommets présents dans le graphe à cet instant, ce qui donne à chaque noeud une probabilité strictement positive d'être connecté. En effet, cette formule diffère de l'originale qui est : $p = \frac{deg(v)}{|E|}$ qui signifie que tout noeud isolé a une probabilité égale à 0 d'être connecté aux autres noeuds du graphe.
La distribution des degrés des noeuds de ce modèle suit une loi de puissance. Il y a très de peu de noeuds fortement connectés, et beaucoup de noeuds faiblement connectés. Nos tests ont portés sur des graphes de taille 5 à 200 (nombre de

noeuds) par pas de 5. Le nombre d'arêtes minimum d'arêtes à ajouter par pas de la génération du graphe a pris les valeurs de 1 à 4, d'où la notation BA1, BA2, BA3, et BA4. Sur la figure 6, on a appliqué l'algorithme MIS sur l'ensemble des graphes décrits précédemment. On note que sur ce type de graphe, cet algorithme se comporte comme s'il possédait une complexité en temps linéaire. Il semblerait que plus la densité du graphe augmente, plus on perd cette linéarité pour commencer à obtenir une courbe exponentielle (une régression non-linéaire sur la courbe BA4 ne donne cependant qu'une fonction de la forme $c \cdot 1,046^x$... avec $c < 1$). On ne dépasse pas 5 secs d'exécution pour les cas les plus difficiles sur une machine de bureau.

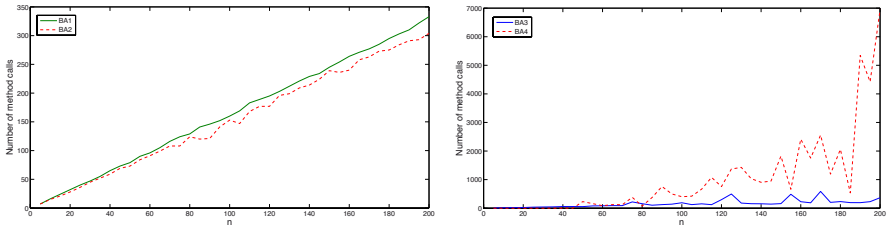
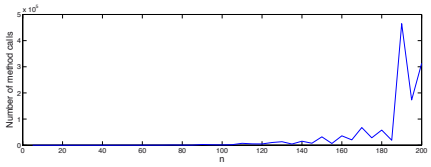


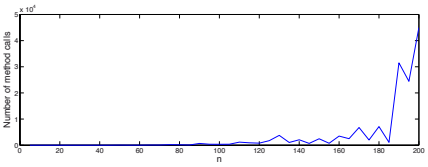
FIGURE 6 – Nombre d'appels de la méthode MIS version FGK09 pour les modèles BA1, BA2, BA3, et BA4.

Pour un nombre de noeuds identiques, le nombre d'appels augmente avec la densité du graphe. Pour information, les densités en moyenne sont de 1,6% pour BA1, 3,4% pour BA2, 5% pour BA3 et 6,8% pour BA4.

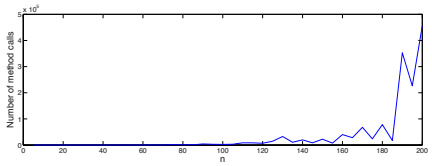
Concernant les algorithmes MVC, les 3 méthodes ont été testées uniquement sur le modèle BA4 (figure 7), les autres modèles ne donnant pas de résultats intéressants. En terme de nombre moyen d'appels de méthode, la méthode Deg-Max + Kernelization permet d'obtenir en moyenne 10 fois moins d'appels que les 2 autres méthodes. On est de l'ordre de la seconde en exécution dans le pire cas.



(a) Degree Branching Strategy



(b) DegMax Branching + Buss Kernelization



(c) Niedermeier Branching

FIGURE 7 – Méthodes MVC appliquées au modèle BA4.

3.1.2 Modèle d'Eppstein

Le modèle Power Law d'Eppstein (EPL) [EW02] possède, comme son nom l'indique, une distribution des degrés suivant une loi de puissance. La différence majeure avec le modèle BA précédent est que le modèle EPL n'est pas un modèle d'accroissement de graphe. Le nombre de noeuds et d'arêtes est fixé dès le départ. L'algorithme de génération part du modèle aléatoire ER, puis redirige des arêtes en pondérant la probabilité d'attacher une arête par le degré du noeud. Plus le degré d'un noeud est élevé, plus la probabilité de reconnecter l'arête à celui-ci sera forte. Ce modèle peut se voir comme une chaîne de Markov apériodique.
Concernant l'algorithme MIS, nous avons cherché la densité qui demandait le plus de nombre d'appels récursifs et il s'est avéré que celle-ci est d'environ 8% (voir figure 8 pour $n = 100$). Néanmoins, la régression non-linéaire de cette courbe (figure 9) ne dépasse pas $1,1152^n$ (légèrement plus que le pire cas du modèle ER qui était de $1,1075^n$). En fait, cela n'a rien d'étonnant, car le modèle EPL est basé sur le modèle ER. La méthode de génération de graphe EPL possède un paramètre qui est le nombre d'arêtes que l'on redirige. Plus on redirige d'arêtes, plus l'approximation de la loi de puissance pour la distribution des degrés est bonne. Dans nos tests avec les graphes EPL, plus la distribution des degrés s'approche d'une loi de puissance, plus le nombre d'appels de la méthode MIS diminue.

Pour ce qui est des algorithmes MVC, ceux-ci se comportent de la même manière que pour le modèle BA (avec $DegMaxKern < DBS < NiedermeierB$).

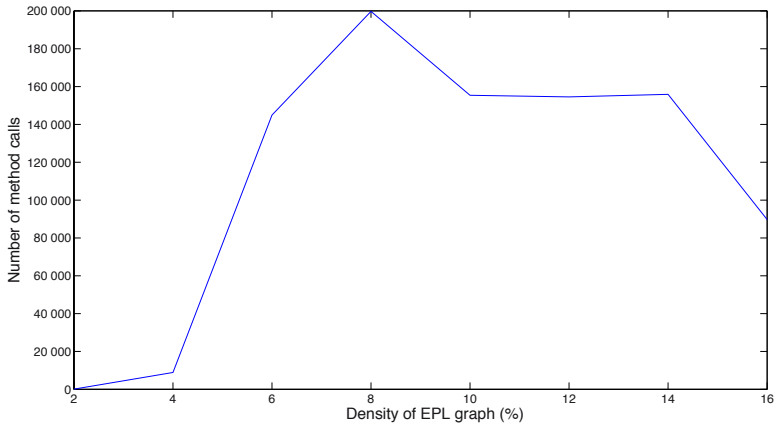


FIGURE 8 – Nombre d'appels de la méthode MIS version FGK09 pour le modèle EPL en fonction de la densité du graphe pour $n = 100$.

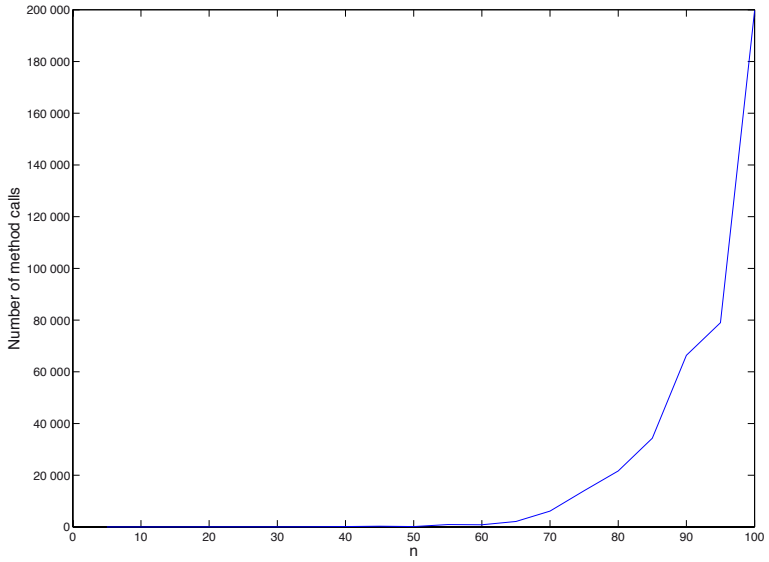


FIGURE 9 – Nombre d'appels de la méthode MIS version FGK09 pour le modèle EPL avec $d = 8\%$

3.1.3 Modèles Small-World

Nous nous sommes intéressés à 2 modèles Small-World : le modèle de Watts-Strogatz [WS98] et le modèle de Kleinberg [Kle00].
Le modèle de Watts et Strogatz (WSSW) possède deux principales propriétés qui sont une longueur moyenne des chemins petite, et un fort coefficient de clustering. La construction d'un WSSW se fait comme suit : on part d'un trellis k-régulier d'anneaux composé de n noeuds, dont on va rediriger chaque arête avec une probabilité p (figure 10). La transition entre le graphe régulier et le small-world se fait pour p entre 0,01 et 0,1.
L'algorithme MIS appliqué à ce type de graphe ne génère que très peu d'appels récursifs (de l'ordre de $2n$). Les algorithmes MVC ont plus de difficultés. Cependant, l'algorithme de Niedermeier se démarque en n'effectuant que très peu d'appels par rapport aux deux autres algorithmes (qui restent tout de même assez éloignés de leur borne théorique) (figure 11). Dans nos tests, k est fixé à 4. Le comportement des algorithmes MVC pour WSSW est $NiedermeierB < DBS < DegMaxKern$.

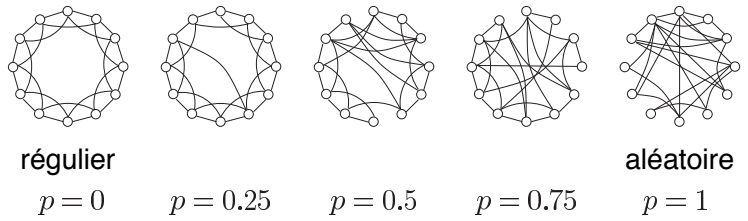
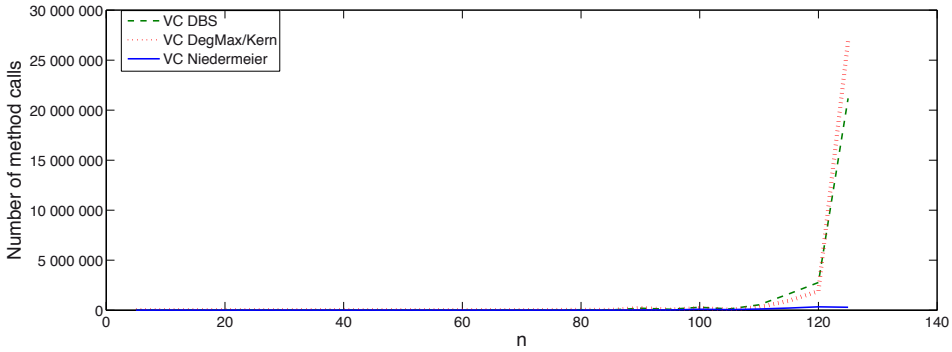
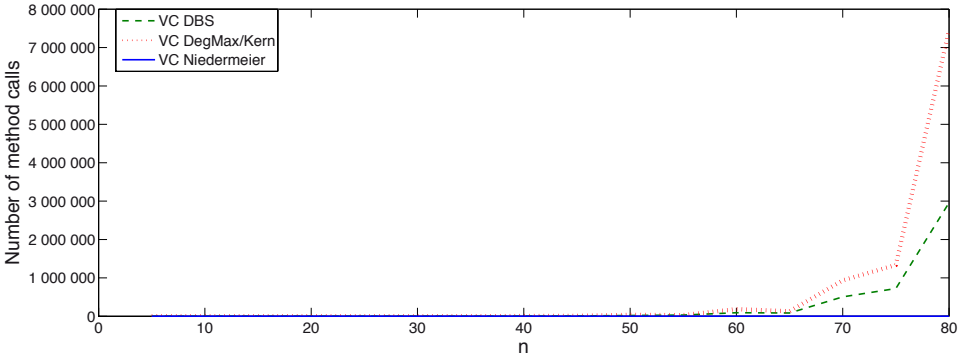


FIGURE 10 – Exemple de small-worlds de Watts et Strogatz



(a) $p = 0.1$



(b) $p = 0.01$

FIGURE 11 – Nombre d'appels des méthodes MVC pour le modèle WSSW avec $p = 0.1$ (a) et $p = 0.01$ (b).

Le modèle small-world de Kleinberg (KSW) est au départ utilisé pour des problèmes de routage dans les réseaux informatiques. Sa construction se fait en prenant comme base un graphe sous forme de grille et en ajoutant des arêtes entre tout couple de noeuds avec une probabilité p.
Concernant les expérimentations sur les algorithmes MIS (figure 13) et MVC (figure 14), on note un pic du nombre d'appels de méthodes pour $p = 0.1$, qui est cependant bien moindre que les résultats obtenus pour le modèle WSSW (en terme de nombre d'appels, on retrouve $DegMaxKern < DBS < NiedermeierB$).

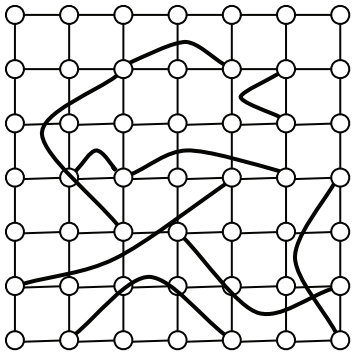


FIGURE 12 – Exemple de small-world de Kleinberg (version non-orientée)

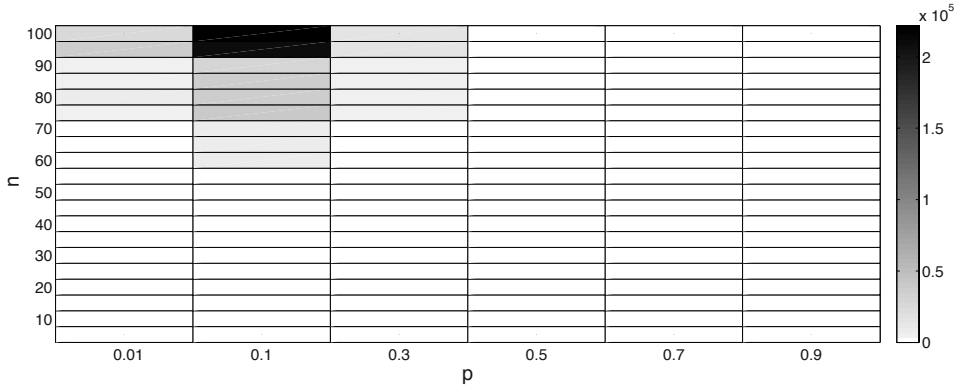
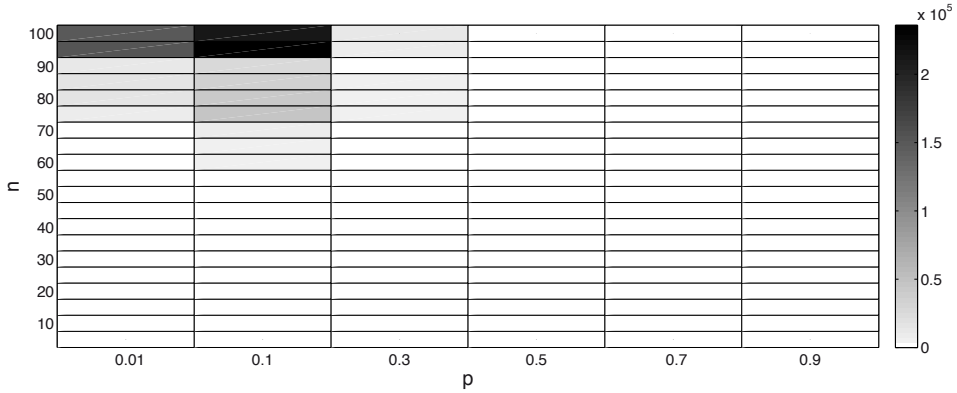
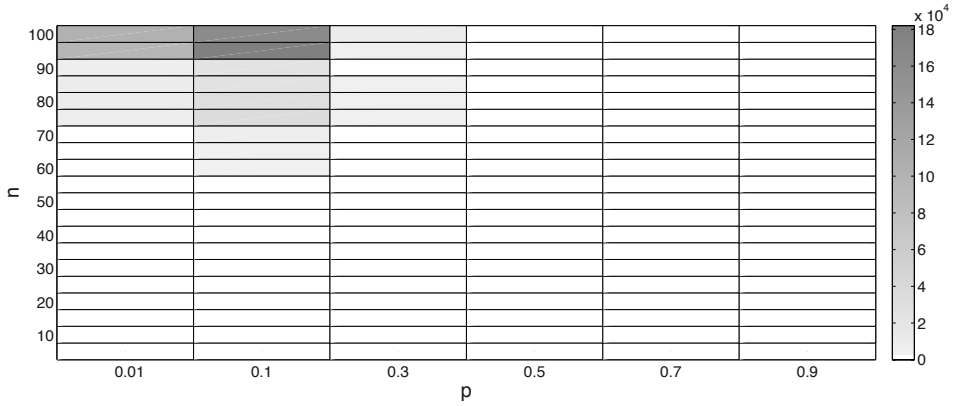


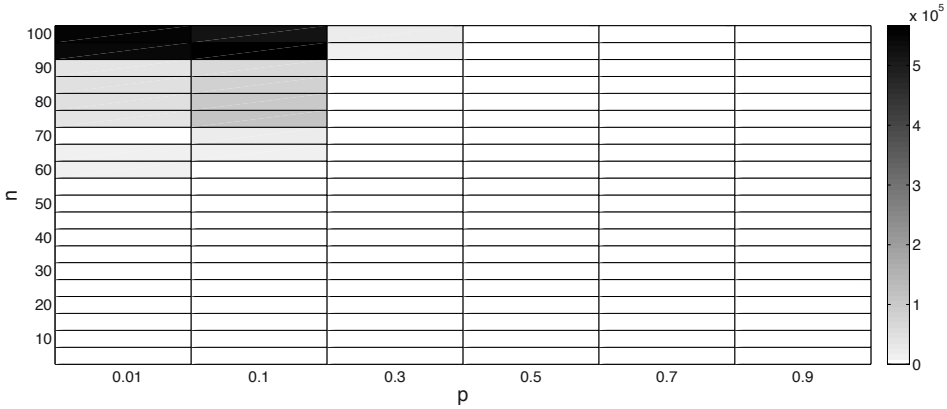
FIGURE 13 – Nombre d'appels de MIS pour le modèle KSW



(a) DBS



(b) DegMax/Kern



(c) Niedermeier

FIGURE 14 – Nombre d'appels des méthodes MVC pour le modèle KSW.

3.2 Graphes non aléatoires

3.2.1 Graphe sous forme de grille

Ici nous avons appliqué les algorithmes MIS et MVC à des graphes sous forme de grille $x \times x$. Avec n le nombre de noeuds atteignant au maximum les 100 noeuds, MIS nécessite en moyenne 1 200 appels ce qui est peu au regard des autres graphes testés. Pour MVC, en nombre d'appels, on obtient $DBS < NiedermeierB < DegMaxKern$. Les résultats sur des grilles toriques sont sensiblement les mêmes.

3.2.2 Graphe sous forme de treillis d'anneaux

Nous nous sommes penchés sur un type de treillis particulier : le graphe trellis k-régulier d'anneaux, avec k pair. Pour $k = 2$, c'est un anneau de longueur n. Pour $k = 4$, le graphe est composé d'un anneau de longueur n et de 2 anneaux de longueur $n/2$. Pour $k = 6$, le graphe est composé de 6 anneaux avec 1 de longueur n, 2 de longueur $n/2$, et 3 de longueur $n/3$. La figure 15 illustre cette construction pour $k = 4$. La définition la plus usuelle [BR91] d'un graphe treillis $L_{m,n}$ est le graphe ligne [Har69] (Line graph) du graphe biparti complet $K_{m,n}$. Le graphe trellis k-régulier d'anneaux est cas particulier du graphe treillis au même titre que les graphes prismes ou graphes carrés par exemple. On se retrouve en face d'un cas particulier d'une sous-classe de treillis qui est en fait un graphe circulant [BH90] du type $Ci_n(1, 2, ..., k/2)$. Nous avons appliqué les algorithmes MIS et MVC sur ces graphes pour $k = 6, 8, 10, 12, 14, 16$ et n de 50 à 100.
L'algorithme MIS se comporte très bien sur ce type de graphe puisqu'il ne dépasse pas les 256 appels pour $n = 100$. Le paramètre k n'a aucune incidence sur le nombre d'appels qui est identique pour chaque valeur de k.
Dans le cas de MVC, on se rend compte que k n'a également aucune incidence sur le nombre d'appels. En nombre d'appels nous obtenons $NiedermeierB << DBS < DegMaxKern$. On obtient avec DBS une complexité "pratique" de 1.19^k (pour 100 noeuds, il peut nécessiter 18 millions d'appels récursifs). L'algorithme DegMaxKern n'est pas du tout efficace en raison de la kernelization ici inutile. On retombe donc sur un algorithme de branchement simpliste. Celui de Niedermeier est en revanche très peu gourmand en nombre d'appels (de l'ordre de $2n$).

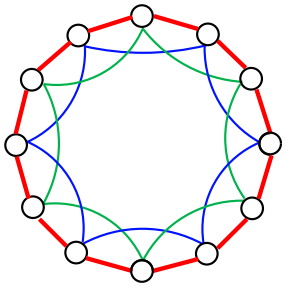


FIGURE 15 – Exemple de trellis 4-régulier d'anneaux

3.3 Graphes aléatoires réguliers

Dans cette partie, nous présentons les résultats obtenus en appliquant les algorithmes MIS et MVC à des graphes réguliers générés aléatoirement d'après le travail de Steger et Wormald [SW99]. Leur algorithme permet de générer de tels graphes en temps polynomial. Voici leur méthode pour générer un graphe k-régulier :

1. Soit G un graphe de n sommets sans arêtes.
2. Soit S l'ensemble des anti-arêtes $\{u, v\}$ de G avec $d(u) \leq k - 1$ et $d(v) \leq k - 1$. Tant que S est non-vide, faire :
   Choisir au hasard une anti-arête $\{u, v\}$ dans S avec une probabilité proportionnelle à $(k - d(u))(k - d(v))$ puis ajouter l'arête $\{u, v\}$ à G.
3. Si G n'est pas k-régulier, aller à l'étape 1, sinon renvoyer G.

Nos tests ont été effectués sur des graphes k-réguliers avec k de 3 à 16 par pas de 1. Concernant l'algorithme MIS, la figure 16 montre le nombres d'appels effectués pour des graphes de 20 à 100 noeuds. La valeur de k pour laquelle le nombre d'appels est maximal est 10 (environ 1 million d'appels). On note ici que ce n'est pas la densité mais bien le degré du graphe qui a de l'importance. La complexité pratique pour $k = 10$ atteint 1.126^n.

Concernant MVC, la même observation apparaît quant au nombre d'appels de méthodes qui forme un pic pour $k = 10$. La figure 17 montre les performances obtenues pour les trois algorithmes étudiés avec $NiedermeierB < DBS < DegMaxKern$ (1.14^k comme borne pratique pour Niedermeier).

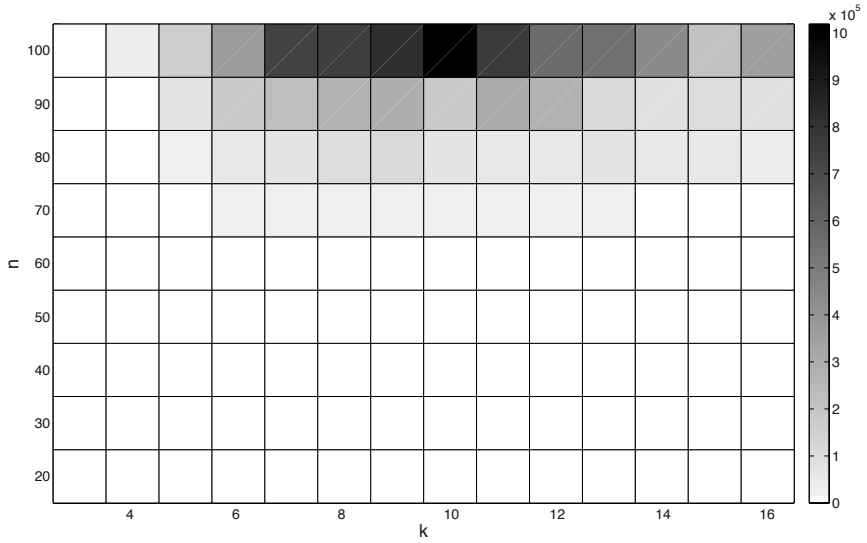


FIGURE 16 – Nombre d'appels de la méthode MIS pour les graphes réguliers aléatoires.

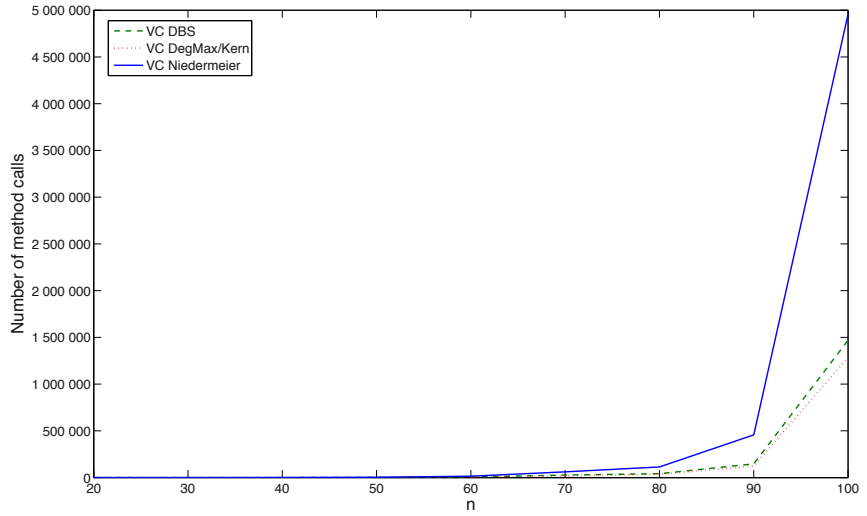


FIGURE 17 – Nombre d'appels des méthodes MVC pour les graphes réguliers aléatoires pour $k = 10$.

3.4 Exploration du voisinage de graphe

L'idée est de trouver une méthode pour construire artificiellement une instance de graphe difficile pour un algorithme donné (Hard Graph). On part d'un graphe aléatoire, et on explore le voisinage du graphe de manière à rendre l'instance à traiter plus difficile pour le problème donné. On exécute les étapes suivantes :

1. On génère un graphe aléatoire ER G.
2. On choisit une arête et on la redirige de manière à maximiser le nombre d'appels de la méthode.
3. On recommence l'étape 2. tant que l'on peut obtenir un nombre d'appels de méthode supérieur à l'ancien, puis on renvoie G.

Evidemment, cet algorithme est coûteux, mais on ne joue que sur la structure du graphe, puisque le nombre d'arêtes et de noeuds ne changent pas. Pour l'algorithme MIS, on avait remarqué que les graphes aléatoires ER qui posaient problème étaient ceux avec $p = 0.2$. Nos tests ont donc été effectués sur des graphes ER avec $p = 0.2$ et $n = 20$. Après plusieurs expérimentations, on obtient un résultat intéressant (voir tableau 2) : on obtient un graphe plus difficile à résoudre pour MIS car on force l'algorithme à passer par des cas qu'il ne peut plus gérer (dans notre cas par exemple, il ne peut plus plier autant de noeuds et doit donc faire des branchements). On reste néanmoins en dessous des résultats obtenus avec des graphes réguliers aléatoires (environ 110 appels de méthode pour $n = 20$).

	ER Graph	**Hard Graph**
Dominance	8	29
Folding	4	1
N[v] branching	0	4
Total	19	91

Table 2 – Nombre d'appels des sous-méthodes de MIS pour $n = 20$.

3.5 Récapitulatif des résultats

Graph type	**MIS** (FGK09)	**MVC** (DBS)	**MVC** (DegMax/Kern)	**MVC** (Niedermeier)
Erdös-Rényi	214 859	197 052	157 695	498 215
Barabási-Albert	402	1 260	358	2 449
Eppstein	199 732	198 291	152 623	560 611
Kleinberg SW	221 774	210 669	163 188	515 313
Watts-Strogatz SW	292	580 044	400 733	178 137
2D Grids	5 456	14 554	31 564	23 299
Ring Lattices Graphs	256	*18 527 881*	$\simeq 2^{67}$	197
Random Regular Graphs	*1 018 213*	1 469 966	1 287 828	*4 957 195*

TABLE 3 – Nombre d'appels de méthode pour MIS et MVC pour $|V| = 100$.

Certains traits de modèles ressortent faisant augmenter le nombre d'appels d'une méthode donnée (voir tabeau 3 avec le maximum d'appels pour chaque méthode en italique). Il semble que des propriétés globales trop générales n'ont pas une incidence franche sur l'efficacité des algorithmes (distribution des degrés, clustering, diamètre...). En revanche, il est clair que l'aspect régulier d'un graphe montre ici toute son importance. Il est d'ailleurs intéressant de noter que la régularité à elle-seule ne suffit pas toujours à proposer une instance difficile pour un problème donné, l'aspect construction aléatoire étant également important (exemple de MIS avec 256 pour les treillis d'anneaux et environ 1 million pour les graphes réguliers aléatoires).
Remarque importante : il faut noter que le nombre d'appels est une chose, mais qu'il y a une part de biais dans ces résultats car on ne sait pas par quelle branche de l'algorithme on passe, certaines étant plus gourmandes que d'autres en complexité polynomiale. Par exemple, pour le cas des graphes aléatoires réguliers, les methodes MVC Niedermeier et DBS sont aussi rapides en temps alors que le nombre d'appels varie de 5 millions à 1,5 millions !

4 Conclusion - Perspectives

La suite de ces travaux consistera à étudier des algorithmes autour des problèmes :

- Coloration (espace polynomial)
- Séparateurs minimaux / cliques maximales potentielles

La recherche de classes de graphes difficiles sera également examiné avec une direction de travail fortement axée autour des graphes réguliers (treillis, graphes prisme, graphes couronne, etc.) et quasi-réguliers (graphe de Barbell, etc.). La génération de Hard Graph fera l'objet d'une étude plus poussée.

Références

[BA99] Albert-Laszlo Barabasi and Réka Albert. Emergence of scaling in random networks. *SCIENCE*, Vol 286 :509–512, 1999.

[BG93] Jonathan F. Buss and Judy Goldsmith. Nondeterminism within p. *SIAM J. Comput.*, 22(3) :560–572, 1993.

[BH90] Fred Buckley and Frank Harary. *Distance in Graphs*. Perseus Books, 1990.

[BR91] Richard A. Brualdi and Herbert J. Ryser. *Combinatorial Matrix Theory*. Cambridge University Press, 1991.

[EW02] David Eppstein and Joseph Wang. A steady state model for graph power law. In *2nd International Workshop on Web Dynamics*, 2002.

[FGK09] Fedor V. Fomin, Fabrizio Grandoni, and Dieter Kratsch. A measure & conquer approach for the analysis of exact algorithms. *Journal of the ACM*, 56(5) :1–32, 2009.

[Har69] Frank Harary. *Graph Theory*. Addison-Wesley, Reading, 1969.

[Kle00] Jon M. Kleinberg. The small-world phenomenon : An algorithmic perspective. In *32nd ACM Symposium on Theory of Computing*, 2000.

[Nie06] Rolf Niedermeier. *Invitation to Fixed Parameter Algorithms*. Oxford University Press, USA, 2006.

[SW99] Agelika Steger and Nicholas C. Wormald. Generating random regular graphs quickly. *Comb. Probab. Comput.*, 8(4) :377–396, 1999.

[WS98] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *NATURE*, Vol 393 :440–442, 1998.