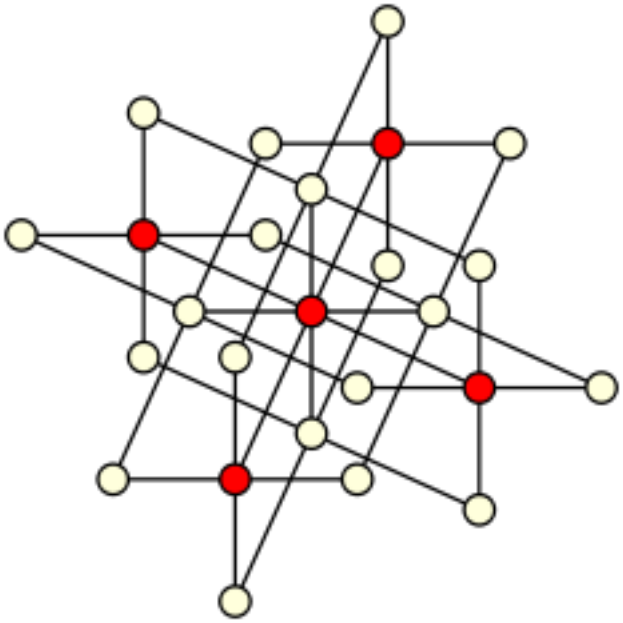
Dissertation sur les algorithmes présentés dans : **On greedy construction of connected dominating sets in wireless networks** de Li, Thai, Wang, Yi, Wan, and Du.



Adrien Miss
Yannick-Alain Couassi-Blé

Rapport de recherche

Master Informatique

Science et Technologie du Logiciel
Université Pierre & Marie Curie
Paris

Octobre 2017

Dissertation sur les algorithmes présentés dans [Li, Thai, Wang, Yi, Wan, and Du. Wireless Communications and Mobile Computing, 2005]

Adrien Miss

Yannick-Alain Couassi-blé

Analyse d'Algorithme et Génération aléatoire – Science et Technologie du Logiciel (STL)

Rapport de recherche – 22 Octobre 2017 – 16 pages

Résumé : Dans ce rapport nous analysons les algorithmes proposés par Li et al. Il propose un algorithme glouton résolvant le problème de l'ensemble dominant connexe avec un ratio de performance de $4.8 + ln5$ de la solution optimale. Nous porterons notre jugement en le comparant à un autre algorithme utilisant un arbre couvrant de Steiner.

Mots-clés : ensemble stable maximal (MIS), ensemble dominant connexe minimum(MCDS), graphe géométrique, algorithme Li et al., algorithme de Steiner

Table des matières

Chapitre 1

Introduction

Nous étudions dans ce rapport les différents algorithmes présentés dans l'article [1]. Les auteurs représentent un réseau sans fil par un graphe non orienté $G = (V, E)$ ou V est l'ensemble des hôtes et E l'ensemble des liens de ce réseau. En plus Ils supposent que la portée maximale de transmission de l'information est identique dans le réseau. De ce fait nous utiliserons des graphes géométriques dans un plan en 2D tout au long de nos travaux. Un graphe $G = (V, E)$ est dit géométrique si et seulement si $xy \in E$ tel que $\forall x, y \in V$, $|xy| < k$ avec $k \in \mathbb{R}_+$. k est appelé le seuil et est fixe.

Le but de l'article est de trouver un ensemble dominant connexe minimal (MCDS) qui est un problème NP-difficile dont la preuve est donnée dans la référence suivante [2]. Un ensemble dominant de G est $X \subseteq G$ tel que $\forall x \in V$ on a $x \in X$ où $\exists y$ tel que $xy \in E$. L'algorithme glouton proposé par les auteurs à un ratio de performance de $4.8 + ln5$ de la solution optimale. Nous présenterons par la suite leur algorithme, l'implémentation et une ouverture vis à vis des autres solutions existantes pour résoudre le probleme de MCDS. On apportera des précisions sur notre méthodologie de génération des instances de tests.

Chapitre 2

Présentation de l'algorithme S-MIS

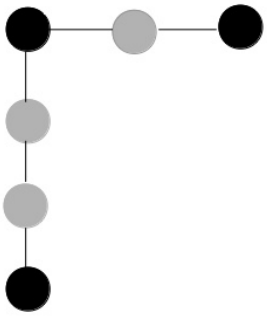
L'algorithme glouton proposé est le **S-MIS**. Il est réalisé en 2 grandes étapes :

1. Construction du MIS respectant le lemme suivant :
   Lemme : $\forall x, y \in MIS,\ k < |xy| <= 2 * k$ avec $k \in \mathbb{R}_+$
2. Application de **l'algorithme de Li et al.** sur le MIS pour engendré le MCDS

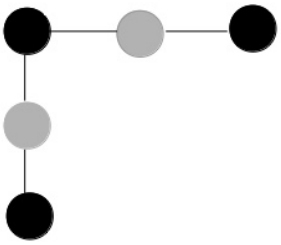
Néanmoins aucune construction de l'étape 1 n'est donnée par Li et al. Mais ils donnent des pistes d'algorithmes qui respectent ce lemme. Donc nous allons implémenter à présent celui de l'article [3].

2.1 Construction du MIS

Un MIS est un ensemble stable maximal qui se définit de manière suivante : Soit $G = (V, E)$, $S \subseteq G$ est stable si et seulement si le graphe induit par S noté $G[S]$ est sans arrêtes. Cette condition est nécessaire pour tout MIS mais n'est pas suffisante pour celui proposé par Li et al. En effet en explicitant le lemme précédent, on en déduit qu'il ne peut avoir au plus un seul sommet n'appartenant pas au MIS entre deux sommets quelconque du MIS. Les figures ci-dessous montrent un exemple de configuration.



un MIS non valide



un MIS valide

2.1.1 Principe et algorithme

Pour simplifier notre implémentation nous présentons ci-dessous les étapes de construction du MIS suivant :

1. On part de $G = (V, E)$ dans lequel on marque tous les sommets en blancs
2. On choisit de manière aléatoire un sommet qu'on marque noir, il est le premier sommet appartenant au MIS
3. Ensuite on marque tous les voisins de ce point en gris (cf : ils ne peuvent pas être dans le MIS)
4. On parcourt chaque point gris en prenant ses voisins comme candidat pour être dans le mis qu'on met dans une liste
5. Et on reprend cette opération tant qu'il reste des sommets à analyser dans la liste
6. On obtient alors notre $MIS = (S, T)$ avec $S \subseteq V$ et $T = \emptyset$

2.1.2 Complexité

On s'intéresse ici à une estimation de la complexité temporelle dans le pire des cas.
On a : $n = |V|$, $k = |E|$

— À l'initialisation : le premier parcourt pour la coloration en blancs coûte $O(n)$ **1**

— Boucle while principal : on prend les points dans cette liste qu'on teste au fur et à mesure avant de les rajouter dans le MIS ou pas. De ce fait on peut tester au maximum n points. Donc elle coûte $O(n)$ **5**

— la boucle **5** comporte des imbrications : le parcourt des voisins d'un point du MIS pour la coloration coûte de même la taille des voisins c'est à dire $O(k)$. Mais cependant pour des questions d'optimisation on peut dès l'initialisation de l'algorithme

calculer une table de hashage qui va d'un point vers la liste de ses voisins. Cependant on réduit le coût en temps constant $O(1)$ **3**

— enfin la recherche des voisins d'un gris correspond à la recherche des voisins des voisins donc le **4** coûte $O(k * k)$

Donc la complexité totale est $O(n * k * k)$. Mais dans notre cas avec les instances de tests sur des graphes géométriques, la recherche de voisin d'un point est beaucoup plus petit que k. On pourrait l'estimer à $\sqrt{k}$ ce qui apporterait une correction d'une complexité totale de $O(n * k)$.

2.2 Algorithme de Li et al.

Leur nouvel algorithme permet donc d'obtenir un MCDS de bonne qualité, mais il doit être appliqué à un jeu de point stable(MIS). À partir de ce moment, il est possible d'appliquer leur algorithme glouton. Dans cet algorithme des couleurs sont attribués aux noeuds, afin de pouvoir les distinguer. Les noeuds noirs correspondent au noeud formant le MIS, à noter qu'il n'y aura aucun changement sur les noeuds de couleur noire. Les autres noeuds de l'ensemble de départ seront tous de couleur gris, avant qu'une partie d'entre eux ne se transforment en bleu. Afin de pouvoir comprendre l'algorithme, il est nécessaire de comprendre ce qu'est un "composant bleu noire" car très peu détaillé dans l'article.

Composant bleu noire : c'est un graphe ayant pour arêtes des noeuds noire-bleu. A noter que les arêtes contenant des noeuds bleu-bleu sont proscrites. Une arête est crée si deux noeuds sont a une distance inférieur au seuil. Le composant bleu noire est le graphe qui découle de ces conditions. Algorithme de Li :

— boucle "for" itérant avec la variable i de 5 à 2 boucle interne au for permettant de parcourir tous les points gris

— si le point gris a une arrête avec au moins i "composant bleu noir" diffèrent alors le point devient bleu.

2.3 Analyse théorique

Nous allons estimer la complexité temporelle de l'algorithme de Li. On a n : le nombre de point du graphe de départ. k est la taille du MIS. La complexité temporelle est donc : $O(4 * ((n-k) * k * (n-k)))$ avec 4 le nombre de tour de la boucle for, $(n-k)$ est le tour de la boucle while dans le pire des cas, en effet il n'y aura pas plus de (n-k) noeud gris

dans l'algorithme. Quand au $k * (n - k)$ il correspond au calcul des graphes, dans le pire des cas le graphe contient tous les noeuds noires (k), mais également tous les noeuds bleus (inférieur à $n-k$). C'est comme ça que nous obtenons notre complexité temporelle. Quand a la complexité en espace, $n * n$ d'espace de point est nécessaire, en effet le parcours des noeuds gris et leurs changement de couleur nécessite seulement une complexité de n. Mais c'est le parcours de graphe qui nécessite une nouvelle collection de point au maximum.

Chapitre 3

Expérimentations et résultats

3.1 Méthodes de génération des bornes de tests

Pour évaluer les algorithmes de Li et al. ainsi que pour les comparatifs, Nous avons effectué une génération pseudo-aléatoire de notre batterie de test. On part d'un cercle conteneur dans lequel on génère nos points en prenant en compte les facteurs ci-dessous :

1. le nombre de points et le seuil qui représente la distance entre 2 points voisins
2. la densité qui représente la distribution des points par élément de surface.
   $densité = \frac{nombre\ de\ points}{Aire\ du\ cercle}$

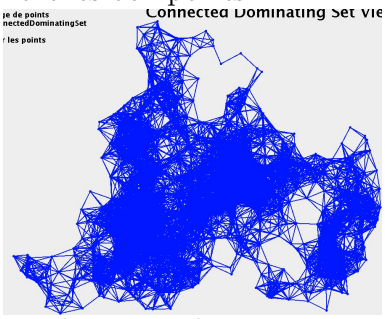
Donnés : rayon=450, seuil=55, nombre d'instances=100

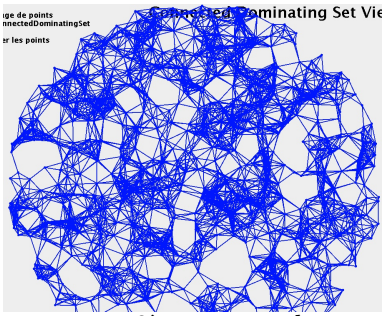
On a effectué une première approche qui consistait à choisir un premier point quelconque dans le cercle et générer le prochain point en étant voisin du précèdent et ainsi de suite jusqu'à attendre le nombre de points voulus. L'avantage de cette méthode est qu'on force la connexité du graphe à la génération mais l'inconvénient c'est qu'on se rend compte très rapidement de la morphologie très touffue (point resserré entre eux dans une région). Ce qui entache le bon jugement des algorithmes.

Par contre la deuxième approche, celle que nous adoptons consiste à choisir le nombre de points en fonction de la densité pour espérer avoir une bonne répartition de ce ceux-ci et essayer de garantir la connexité. On teste les algorithmes sur 100 instances par catégorie de nombre de points. Et selon nos tests à partir de 800 points on à des graphes corrects. Néanmoins il peut arriver qu'il reste 1 ou 2 points isolés (qui seront ignorés). Ce qui n'entache pas le bon jugement des comparaisons.

NB : On peut améliorer ce problème en écrivant un algorithme de détection de connexité

du graphe ; et si ce n'est pas le cas de rassembler les composantes connexes en tirant aléatoirement les bon points.


1ère approche


2ème approche

3.2 Premier résultat

Ci-dessous nous avons le résultat de la moyenne de l'algorithme de S-MIS sur les 100 instances par catégorie de nombre de points.

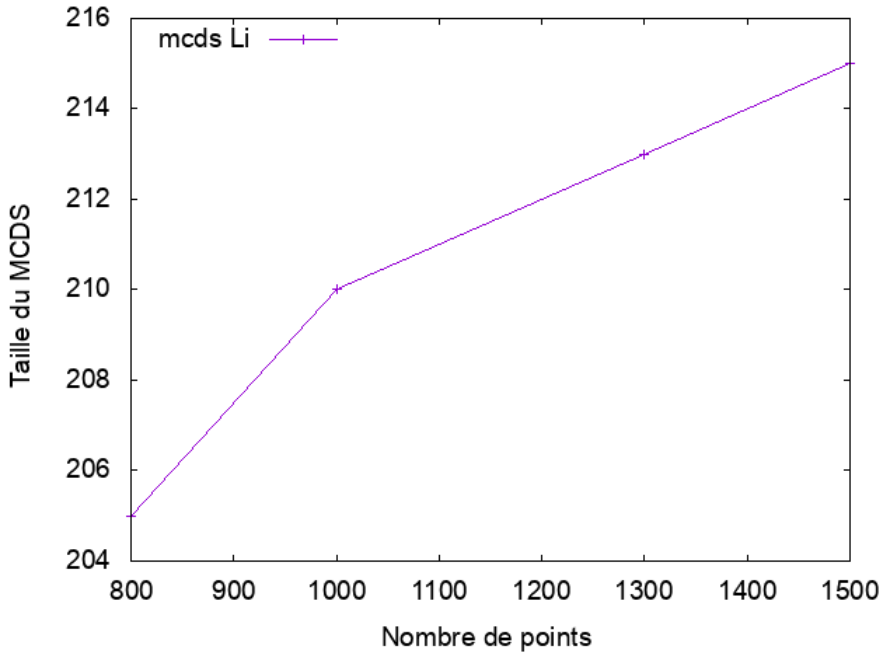


FIGURE 3.1 – Taille du CDS en fonction du nombre de points

On constate que la taille du MIS obtenue est intéressant et évolue peu même avec des nombres de points importants. Nous regarderons plus en détails par la suite le temps d'exécution par rapport à l'application de Steiner sur le MIS.

Chapitre 4

Algorithmes implémentés

4.1 Algorithme de Li : S-MIS

L'algorithme de Li fonctionne en prenant en entrée un échantillon de points représentant le MIS. On attribut une couleur aux points en créant un nouvelle objet PointAvecCouleur, ce dernier possède une énumération comportant sa couleur. On stocke tous les points colorés dans une ArrayList afin de pouvoir y accéder facilement. Nous parcourons points gris et nous comptons le nombre de graphe lié à ce point. Le calcul du nombre de graphe lié à un noeud gris s'effectue en plusieurs étapes :

1. on calcul les noeuds noirs voisins du point gris
2. chaque point est placé dans une ArrayList différentes
3. on stocke ces ArrayList dans une HashMap avec un accès par leur point noir
4. pour chaque point voisin on calcul récursivement son graphe, sauf si ce point est inclus dans un autre graphe, dans ce cas on supprime l'ArrayList de la HashMap
5. on retourne le nombre d'ArrayList contenu dans la HashMap, ce qui correspond bien au nombre de graphe.

Si le nombre de graphe voisin est suffisant, le noeud change de couleur et devient bleu. L'algorithme de Li retourne les points colorés en bleu.

4.2 Algorithme : "Heuristique de Steiner"

L'algorithme dont nous allons parlé est celui implémenté lors du cours d'ALGAV en Master 1 qui est aussi un problème NP-difficile.

Le principe est le suivant : étant muni de P points dans le plan, un arbre A pour P est un ensemble de chemins connectés tel que tous les points soient reliés. A est dit de Steiner si sa longueur totale est minimale.

Cet algorithme est réalisé en plusieurs étapes :

1. *calculShortestPaths()* calcul le plus court chemins entre deux points
2. *KruskalList()* : permet de retourner la manière la plus courte de relié deux hitPoint(dans notre cas il s'agit de notre MIS).
3. Ensuite on stock les points du MIS ainsi que les plus court chemin permettant de les relier dans une variable p
4. on appel ensuite la fonction *Kruskal()* qui va nous permettre d'obtenir l'arbre couvrant minimal.

Notre algorithme implémenté a une complexité temporelle d'environ $O(n^3)$.

4.3 S-MIS vs Steiner

On réalise une étude qualitative des performances du S-MIS face à celui de Steiner après application sur le même MIS dans les même conditions.

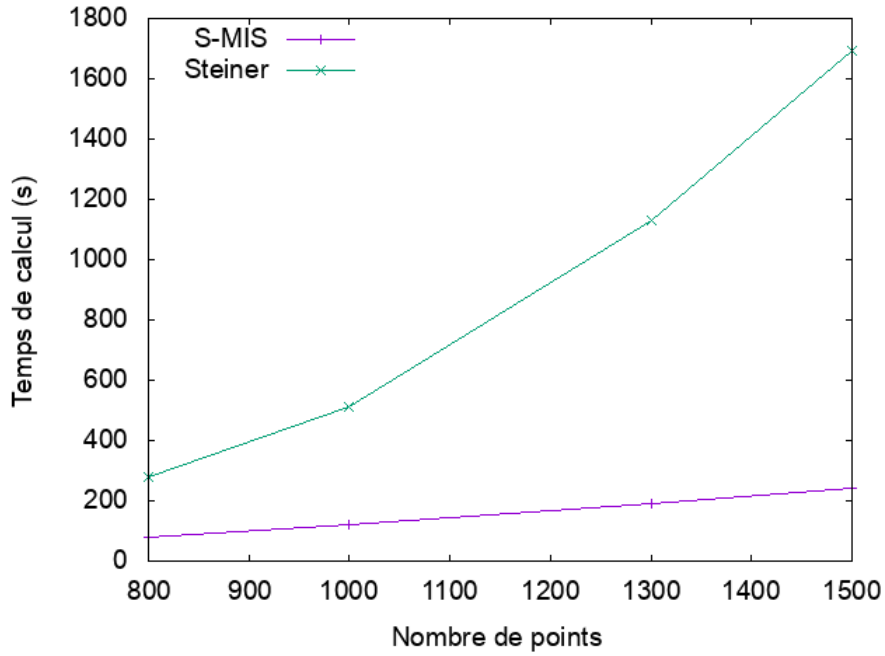


FIGURE 4.1 – Comparaison temporelle S-MIS et Steiner

On s'aperçoit directement que le S-MIS est fatalement plus rapide que le Steiner en

temps d'exécution. Ce qui est justifié et conforme à la classe de complexité que nous avons évalué théoriquement. La courbe de Steiner s'approche bien d'une fonction en n^3. Par contre celle de S-MIS montre clairement une fonction linéaire alors qu'on l'avait évalué plus proche d'une quadratique (n^2). À ce constat on peut émettre l'hypothèse selon laquelle le calcul des voisins se réalise de manière assez spontané dans le S-MIS.

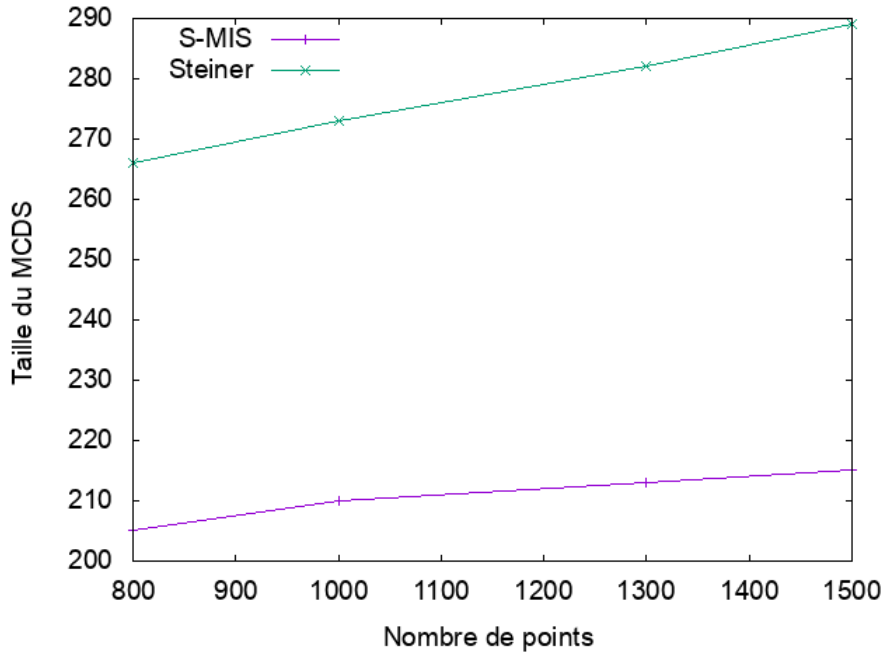


FIGURE 4.2 – Comparaison de la taille du CDS entre S-MIS et Steiner

Ici de même on constate clairement que la taille du CDS obtenue avec steiner augmente fortement avec le nombre de point dans le nuage. Tandis que celui du S-MIS est assez stable.

Chapitre 5

Appréciations, améliorations et critiques de l'algorithme de Li

Nous avons trouvé l'algorithme de Li intéressant pour plusieurs raisons, tout d'abord sa complexité temporelle se situe autour des $O(n^2)$, ce qui est intéressant quand on regarde la difficulté du problème. De plus son implémentation est d'une relative simplicité, même si le comptage du nombre de graphe demande de la réflexion. Cet algorithme est bien meilleur que celui de Steiner à tous les niveaux, on obtient des scores meilleurs et des temps de calcul records. Néanmoins un des inconvénients est que le S-MIS force le type de MIS sur lequel il veut s'appliquer. Alors que lorsqu'on effectue un Steiner sur un autre type de MIS, on obtient des résultats assez proche. On pourrait rendre le S-MIS encore meilleur en effectuant par exemple un localSearching sur le CDS obtenue afin de réduire sa taille. Celui que nous avons implémenté mettait énormément de temps à calculer c'est pourquoi nous ne l'avions pas mentionné dans les tests. Une autre amélioration est de mettre de l'aléatoire dans le calcul du MIS par exemple, ce que nous faisons directement à l'initialisation après coloration en blancs des points. Ce qui permet de résoudre le choix aléatoire du premier point du MIS au lieu de faire un algorithme probabiliste d'élection par exemple. Enfin le graphe que l'on obtient avec cet algorithme possède des cycles, il serait donc intéressant d'en éliminer une partie.

Chapitre 6

Conclusion

L'article que nous avons étudié a été publié en 2005, la méthode qu'ils ont proposé était innovante au moment de sa publication. Mais Aujourd'hui il existe de nombreux articles proposant de nouveau algorithme sur les "Connected Dominating Sets". Il serait donc intéressant d'implémenter un ou deux algorithmes supplémentaires afin déterminer les progrès réalisé ces dernières années. La $4.8 + ln5$-approximation que nous donne le S-MIS est en réalité quasi-linéaire et est une bonne solution pour la résolution d'un problème NP-difficile. Le titre d'algorithme glouton donné par les auteurs est a première vue septique mais après toutes les analyses que nous avons fait on peut conclure que c'est une solution intéressante qu'on peut privilégier dans des domaines concret d'application.

Bibliographie

[1] Yingshu Li, My T. Thai, Feng Wang, Chih-Wei Yi, Peng-Jun Wan and Ding-Zhu Du (2005), **On greedy construction of connected dominating sets in wireless networks**, Wirel. Commun. Mob. Comput. 2005 ; pp 927—932.

[2] Clark BN, Colbourn CJ, Johnson DS. (1990), **Unit disk graphs**, Discrete Mathematics 1990 ; 86 :pp 165—177.

[3] Wan P-J, Alzoubi KM, Frieder O. (2002) **Distributed construction of connected dominating set in wireless ad hoc networks**, IEEE Infocom 2002, New York, NY, USA, June 2002.