Master

Sciences et Technologies du Logiciel

Implementations du problème "Connected Dominating Set"

Réalisé par :
Thamazgha SMAIL

Contents

List of Figures

List of Tables

Mots clés
Connected Dominating Set, Minimum Independant Set

1 Introduction Générale

Le concept de la dominance dans un graphe a plusieurs interets dont la prise en charge du broadcast et le multicast dans les réseaux sans fils, en sachant que ces derniers sont deux méthodes de communication populaires dans les travaux sur ce type de réseaux (1)
Pour cela, les hotes dans un réseau sont representées par des points dans un graphe, et les liens entre ces derniers par des arêtes entre ces points, un reseau sans fil est donc un graphe.
Nous savons que le broadcast et le multicast sont pris en charge par le backbone du réseau, ce dernier pourra donc être modelisé par un ensemble dominant connexe (1) . Pour cela, l'entrée des algorithmes présentés dans ce document est un graphe G=(V,E) tel que V est l'ensemble des nœuds du graphes et E l'ensemble des aretes entre ces derniers, et la sortie sera évidement l'ensemble S de nœuds qui compose l'ensemble dominant connexe.
Dans la plupart des algorithmes de construction de CDS, un mécanisme de coloration est utilisé où initialement tous les nœuds sont blancs, un dominant est coloré en noir et un dominé est gris (1).

2 Analyse et présentation de l'existant

Dans ce qui suit on notera CDS, l'ensemble dominant connexe, S l'ensemble de points de ce dernier et MIS pour Minimum Independant Set.
Dans ce papier, les auteurs présentent plusieurs algorithmes pour construire ce fameux CDS, dont:

2.1 Problématique

En réseau, il est très interéssant de connaitre l'ensemble dominant connexe minimum (MCDS en anglais) car ce dernier permet d'envoyer un message à tous les noeuds du graphe en un minimum de temps grace à la connexité de l'ensemble et sa taille minimale, il suffit donc de passer le message au premier point de l'ensemble dominant connexe, celui ci le passe aux autres points de ce dernier, les points du MCDS transmettent à leur tour le packet aux points restants.
Cependant, calculer un tel ensemble est NP-Difficile (1) mais plusieurs heuristiques ont été proposées pour obtenir cet ensemble là, dont l'article de Yingshu Li et My T. Thai qui sera étudié plus en détails dans les sections qui suivent.

2.1.1 Algorithme Guha et Khuller :

ils ont proposé un premier glouton qui consiste à construire le CDS à un nœud, puis l'espace de recherche pour le prochain dominant est limité aux dominés actuels et le CDS se développe jusqu'à ce qu'il n'y ait plus de nœuds blancs (1)
Ils n'ont par contre pas spécifié la démarche à suivre afin de choisir le prochain

dominant, une solution possible est de choisir à chaque fois le noeud qui a le plus de voisins en faisant attention aux noeuds qui existent dèja dans S, la condition d'arrêt sera comme dans la plupart des cas, la couverture de tous les sommets du graphe et la connexité de S.
Puis ils ont proposé un deuxième algorithme qui consiste à determiner tous les dominants possibles puis connecter ces derniers à travers quelques nœuds intermédiaires. l'heuristique utilisée pour determiner les dominants n'a pas été précisée dans ce papier, une solution possible est de construire le MIS comme suit : les dominants sont des nœuds non-adjacent c-à-d on parcourt l'ensemble de nœuds de départ, si le nœud courrant n'a une arête avec aucun des nœuds de l'ensemble S, initialement vide, alors on l'ajoute à S, ainsi on pourra passer à la deuxième étape qui est d'utiliser des nœuds intermediaires pour connecter les élements de l'ensemble S, une solution possible pour choisir ces derniers est de prendre les voisins des dominants jusqu'à ce que tous les sommets soient couverts et que S soit connexe.

2.1.2 Algorithme de Wu et Li :

Le principe de cet algorithme est le suivant : Si un nœud a deux voisins non connectés, il devient un dominant, Le CDS généré est facile à maintenir. son PR est en O(n), mais sa taille est grande (1).
cet algorithme ne couvre pas l'intégralité des nœuds du graphe, des nœuds qui sont isolés ou qui ont seulement un seul voisin peuvent exister, il faut donc les ajouter au CDS, pour qu'il couvre tous les nœuds, mais il reste quand même pas optimale.

2.1.3 Algorithme de Alzoubi et al.

Leur première approche consiste à construire un arbre couvrant, puis chaque noeud dans l'arbre est étiqueté comme un dominant ou un dominé. Ils ont ensuite remarqué la difficulté de la maintenance du CDS par ce dernier et ont conçu un deuxième algorithme localisé à deux phases réparti qui est bon à l'entretien en général. Un MIS(Maximal Iindependent Set) est généré de manière distribuée sans construire un arbre ou sélectionner un leader. Une fois qu'un nœud sait qu'il a le plus petit ID dans son voisinage à 1 saut, ce nœud devient un dominant. [1]

2.1.4 Algorithme A :

Et enfin, les auteurs ont expliquer leurs propre algorithme qui est plus performant que les algorithmes précedement cités.
Le principe est le suivant : commencer par faire le MIS de notre graphe, marquer tous les noeuds du MIS blancs et le reste des somments en gris et enfin changer quelques somments gris en bleu, ces sommets bleux nous servirront de noeuds

intermédiaires pour rendre le MIS connexe. autrement dit, il faudra détécter les noeuds dominés par plussieurs dominants (de 2 à 5), ces points là seront colorés en bleu.

Dans cet algorithme, deux couleurs (blanc et noir) pouvaient mieux representer le problème, car en réalité, nous avons que deux types de noeuds (dominant et dominé), donc les noeuds blancs et gris auraient suffit. Puis lors de la séléction des noeuds bleux, des noeuds qui n'ont aucun interêt sont ajoutés à l'ensemble S, Nous pouvont résoudre ça en supprimant ces noeuds là de S.

Les resultats de cet algorithme dépendent de la façon dont est construit le MIS.

3 Algorithmes implémentés :

La présente section consistera à présenter les heuristiques implémentées afin d'améliorer la taille du CDS. les algorithmes suivants ne marcheront pas dans le cas ou le graphe n'est pas connexe parce que le but est de construire un CDS, ceci n'est pas possible si même le graphe n'est pas connexe.

3.1 MIS-Steiner

3.2 MIS-BlackBlue

Le principe de cet algorithme est de construire le MIS d'une façon différente que la précedente, qui consiste à : un noeud du graphe appartient à S si seulement si : il n'a pas de voisins ou bien les éléments de S ne dominent pas tous ces voisins.
Puis connecter les noeud du MIS à l'aide de la méthode de l'article qui consiste à parcourir les sommets du graphe et à chaque fois qu'on tombe sur un noeud qui a au moins deux et au plus cinq voisins dans S, nous l'ajoutons à ce dernier.
Le fait d'imposer ces deux contraintes aux noeuds à ajouter est due à :
On sait qu'un noeud quelconque a au plus cinq voisins dans S (1), et un noeud nous permettra de connecter deux sommets dans l'ensemble S si et seulement s'il a au moins deux voisins appartenant à ce dernier. Les noeuds qui répondent à ces deux contraintes nous permettront de relier les noeuds de S.
Décidement, cette méthode donne de meilleurs résultats que la précédente grace à l'heuristique appliquée à la construction du MIS et de la suppression des sommets inutiles qui consiste à supprimer de S dont la suppression ne change rien à la connexité (cf. méthode connected)de S et à la couverture de tous les noeuds (cf. méthode tout-le-monde-est-domine), voici un exemple d'execution de ce dernier.
Un exemple d'exécution (sur test1) est montré dans la figure qui suit .

Le principe de cet algorithme est de construire dans une première phase le MIS en selectionnant les points qui ne sont pas connectés i-e : dès qu'un noeud n'est adjacent avec aucun des éléments de S, initialement vide, nous l'ajoutons à ce dernier, puis dans une deuxième passe, connecter les éléments de S en utilisant l'algorithme de Steiner, le score de cette méthode avec le fichier test1 (contenant 1000 points) est comme sur la figure suivante.

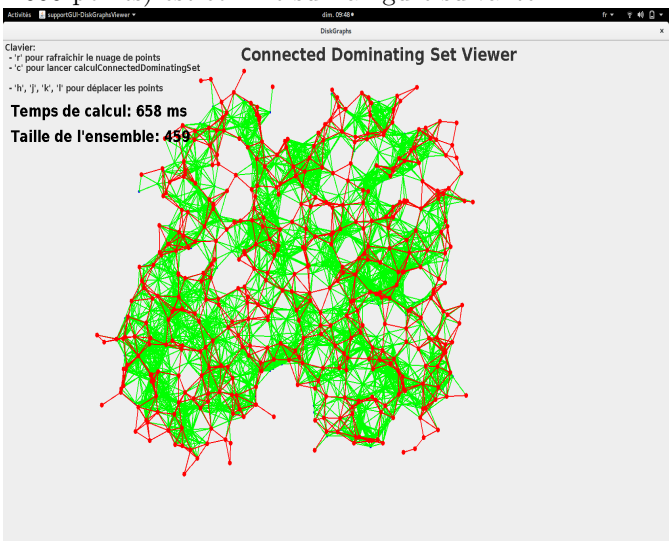

Figure 1: Exemple d'éxecution de MIS-Steiner sur test1.

3.3 Meilleur-Score

Cet algorithme consiste à calculer le MIS avec la même méthode que MIS-BlackBlue, car visiblement elle aide à chosir les bons points de S et améliore donc le score.
Ensuite les sommets de S sont reliés à l'aide de l'algorithme de Steiner.
le score de cette méthode avec le fichier test1 (contenant 1000 points) est comme sur la figure suivante.

3.4 Comparaison entre les algorithmes

Les tests présentés dans le tableau suivant sont fait sur la même instance de points (nommée test1 dans le projet). elle contient 1000 points

Visiblement, MIS-Steiner prends beaucoup de temps pour un nombre de points assez grand, ce qui n'est pas intéressant dans notre cas, surtout que MIS-BlackBlue nous donne un ensemble S réduit en moins de temps que le premier algorithme. Le score de points est amélioré dans le dernier algorithme pour un temps plus important.
Nous pouvons voir le lien entre le temps de calcul et le score en nombre de

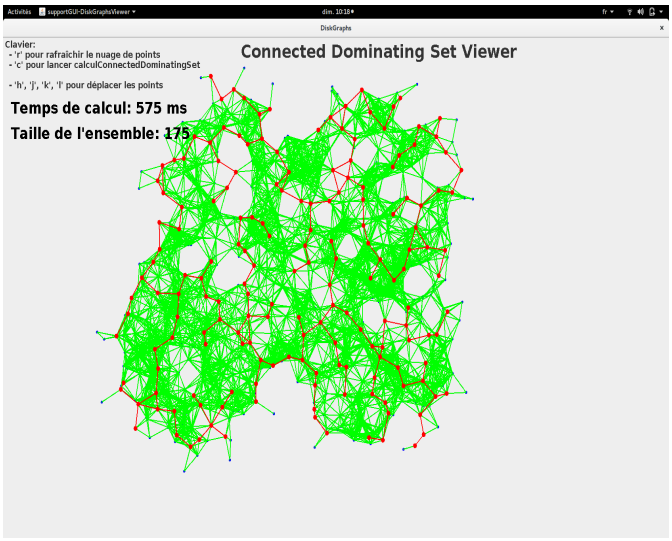

Figure 2: Exemple d'éxecution de MIS-BlackBlue sur test1

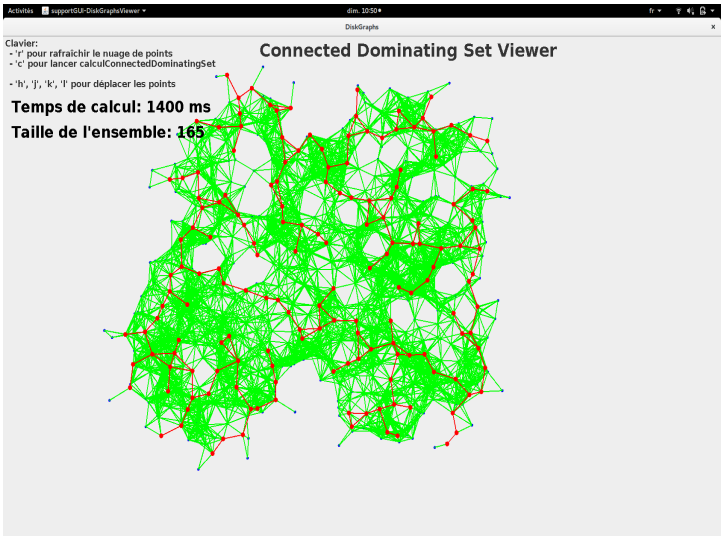

Figure 3: Exemple d'éxecution de Meilleur-Score sur test1

points dans S dans le graphique qui suit :

Algorithme	Temps de calcul (ms)	Nombre de points de S
MIS-Steiner	658	459
MIS-BlackBlue	575	175
Meilleur-Score	1400	165

Table 1: Comparaisons entre les algorithmes

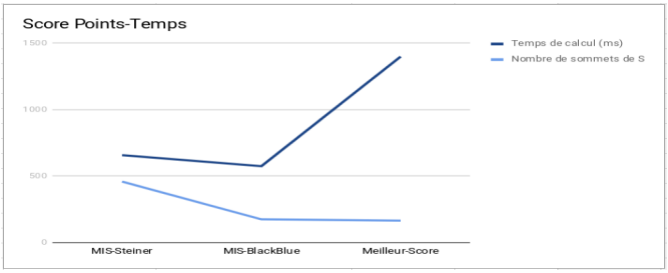

Figure 4: Comparaison

Nous remarquons sur le graphique que le temps de calcul augmente très vite comparé au nombre de points qui ne se réduit pas aussi vite que ça, cela est évidement due au temps d'éxecution des heuristiques.

3.5 Comparaison entre les instances de points avec l'algorithme Meilleur-Score

Les tests suivants ont été fait sur plusieurs instances de points afin de comparer le résultat de l'algorithme "Meilleur-Score" quand la position des points change. Nous avons donc cinq exemples de graphes dont le nombre de points total est égal à 1000, ces derniers ont été obtenus en rafraîchissant le nuage de points du code fourni. la dernière instance contient les points du fichier "test5" plus 21 autres points ajoutés manuellement afin d'observer ce qui est affecté quand le nuage de points augmente.

Instance de points	Nombre total de points	Temps de calcul (ms)	Nbre de points de S
test1	1000	1400	165
test2	1000	1194	158
test3	1000	1215	165
test4	1000	1187	159
test5	1000	1189	157
test6	1021	1258	157

Table 2: Résultats des tests.

Le résultat de ces tests est plus visible sur le graphique suivant :

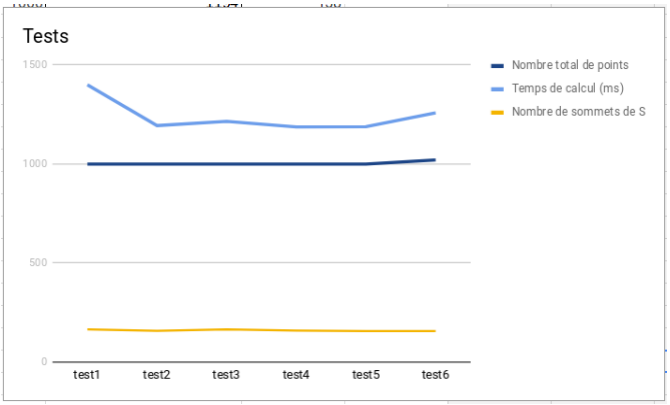

Figure 5: Comparaison entre différentes instances de points avec l'algorithme Meilleur-Score.

Nous remarquons que le temps de calcul est en relation directe avec le nombre de points total du graphe, ce qui est normale vu qu'il s'agit de parcourir les points et de faire des calculs. En revanche, le nombre de sommets dans S reste plus ou moins constant du moment que les points ajoutés ne soient pas très éloignés du nuage de points de départ.

4 Conclusion

Nous avons essayé dans ce projet de reproduire les résultats de l'article. Nous avons été contraints de faire des choix, notamment sur la construction du MIS et clairement le resultat final dépends énormement des points qui compose ce dernier mais aussi des heuristiques utilisées pour les relier.
comme nous pouvons le voir sur la figure 3, il reste des améliorations à faire, notament sur les cycles de trois points apparents. Une perspective serai de gérer les points isolés en rajoutant des points au graphe pour que ces derniers soient connectés au reste de points du graphe ou encore faire attention à la complexité temporelle et spatialle des algorithmes.

References

[1] Y. Li, M. T. Thai, F. Wang, C.-W. Yi, P.-J. Wan, and D.-Z. Du, "On greedy construction of connected dominating sets in wireless networks: Research articles," *Wirel. Commun. Mob. Comput.*, vol. 5, no. 8, pp. 927–932, Dec. 2005. [Online]. Available: http://dx.doi.org/10.1002/wcm.v5:8