Rapport de projet AAGA

Ensemble Dominant Connexe

Thierno BAH - 3408625

Encadrant : Binh-Minh Buixuan

Sommaire

Introduction

Etant donné un graphe **G=(V, E)** où **V** est l'ensemble des noeuds du graphe et **E** l'ensemble des arêtes, un ensemble dominant de **G** est un sous-ensemble **D** de **V** tel que tous les noeuds de **D** permettent d'atteindre tous les noeuds de **V**.

Un ensemble dominant connexe (ou CDS) est un ensemble dominant dont tous les noeuds sont reliés par un chemin, il peut y avoir plusieurs ensembles dominants connexes par graphe.

Nous allons nous intéresser à la recherche du plus petit ensemble dominant connexe.

La recherche de cet ensemble (le plus petit) est un problème NP-Complet, il n'existe donc pas d'algorithme capable de le trouver en un temps raisonnable.

Dans ce rapport nous allons voir deux méthodes qui nous donnent une solution approchée.

Dans un premier temps, nous allons voir l'algorithme de Li et al puis une deuxième méthode un peu plus naïve, nous allons voir comment ont été implémenté ces deux méthodes et voir les résultats obtenus avec celles-ci sur des graphes de différentes tailles.

Algorithmes

Algorithme Li et al

L'algorithme de Li et al est un algorithme glouton qui nous permet de trouver un ensemble dominant connexe, cet algorithme se déroule en deux étapes:

1. Construction d'un ensemble stable maximum
2. Connection des points de l'ensemble stable pour le rendre connexe en ajoutant des noeuds spéciaux (les noeuds de Steiner)

Construction d'un ensemble stable

Un ensemble stable (ou MIS) sur un graphe **G=(V,E)** est un ensemble dominant dont toutes paires de sous-ensembles complémentaires a exactement une distance de deux sauts.

Pour pouvoir construire un tel ensemble, je me suis inspiré d'une méthode qui était abordé dans l'article de Li et al.

En "colorant" les noeuds pour connaître l'état du graphe, j'ai implémenté un algorithme glouton qui me permet de trouver un ensemble stable.

Les noeuds de couleur **noir** sont les noeuds qui font partie de mon ensemble stable, les **gris** vont être ceux qui sont dominés par les noeuds noir et les noeuds **blancs** sont ceux qui ne sont pas encore traités par l'algorithme.

Au début de l'algorithme je colore tous les noeuds en blanc, ensuite parmi ces noeuds blancs, je cherche celui qui a le plus de voisin et je le colore en noir et tous ses voisins en gris.

Ensuite tant qu'il me reste des noeuds blancs (non dominés) je vais à la recherche d'un noeud blanc qui a un voisin gris. Dès que j'en trouve un je l'ajoute à mon ensemble dominant en le coloriant en noir.

A la fin j'ai un ensemble stable, il n'y a aucun noeud noir qui est voisin d'un autre noir ou deux gris qui se suivent entre deux noirs mais exactement un.

Voici un exemple pour illustrer le fonctionnement de l'algorithme :

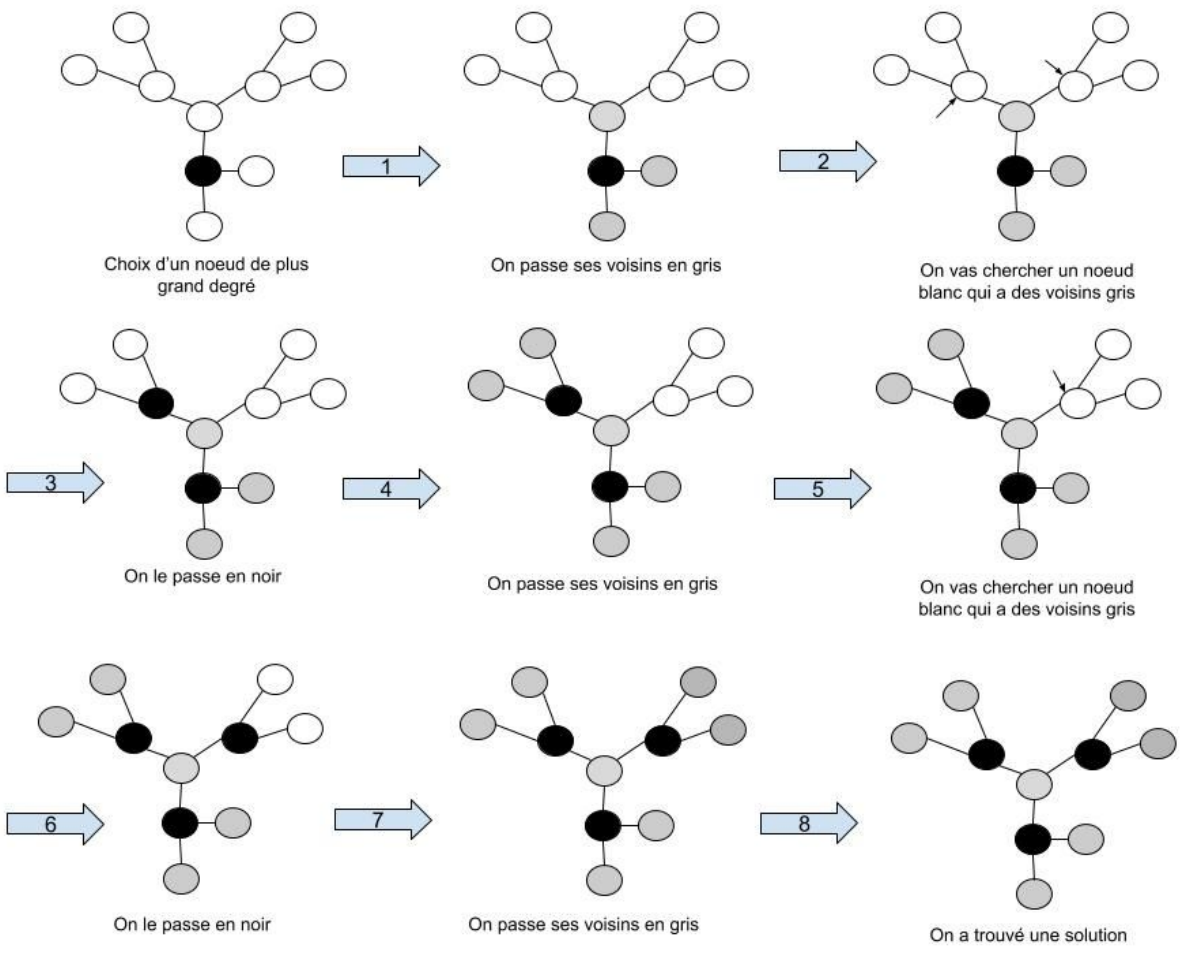

Figure 1 : Schéma du déroulement de l'algorithme pour trouver un ensemble stable

Dans cet exemple l'ensemble trouvé est de taille 3 et il est minimum mais si le premier point qui avait été choisi était celui du milieu nous aurions trouvé un ensemble de taille 7.

Construction de l'ensemble dominant connexe avec les noeuds de Steiner

L'algorithme qui nous permet de construire l'ensemble dominant connexe à partir de l'ensemble stable est décrit dans l'article de Li et al:

INPUT : Un ensemble dominant stable
OUTPUT : Un ensemble dominant connexe
début :
for i = 5; 4; 3; 2 do:
while there exists a grey node adjacent to at least i black nodes in different black-blue components do:
change its color from grey to blue
return all blue nodes + all black nodes
fin

Pour implémenter cet algorithme il a fallu trouver un moyen de représenter les "*black-blue components*" et de facilement les identifier et les "modifier".

Pour implémenter ces composants, j'ai ajouté un attribut en plus à chaque noeud (en plus de sa couleur).

Les noeuds ont un attribut "*composant*" qui est l'identifiant du composant auxquels ils appartiennent, et tous les noeuds qui ont le même identifiant sont dans le même black-blue component.

En modifiant ce composant on peut facilement ajouter ou enlever un noeud à un black-blue component.

Pour faire une union entre deux black-blue components il suffit de prendre un nouvel indice et l'appliquer à tous les noeuds dans les deux black-blue components.

Dans mon implémentation de cet algorithme au début aucun noeud n'est dans un composant black-blue, ils ont tous l'indice 0 pour l'attribut "composant'.

Je cherche tous les noeuds gris et je les ajoute dans une liste.

Ensuite pour les différentes valeurs de *i*, je vais parcourir ma liste de noeuds gris et si je tombe sur un noeud gris qui a plus de *i* voisins je regarde qui parmis ses voisins est noir.

S'il a plus de *i* voisins noirs qui sont tous dans des black-blue components différents ou qui ne sont dans aucun black-blue components (c'est-à-dire l'identifiant composant est à la valeur par défaut 0) je vais à l'étape suivante sinon je continue de chercher un noeud gris qui remplit ces conditions.

Je regarde si je suis dans le cas où les voisins noirs ne sont dans aucun black-blue components, si c'est le cas c'est très simple je mets mon noeuds gris en bleu (je l'enlève de la liste des gris), je crée un nouvel indice et je change l'attribut composant des *i* voisins noirs en y mettant le nouvel indice que j'ai créé.

Si je suis dans le cas où j'ai des noeuds noirs dans différents composants black-blue je vais faire l'union des différents black-blue components comme je l'ai expliqué plus haut et passer le noeud gris en bleu.

A la fin je renvoie mes noeuds bleus et mes noeuds noirs, j'ai un ensemble dominant connexe.

Voici un schéma qui illustre le déroulement de l'algorithme sur l'ensemble stable que nous avions trouvé à la figure 1:

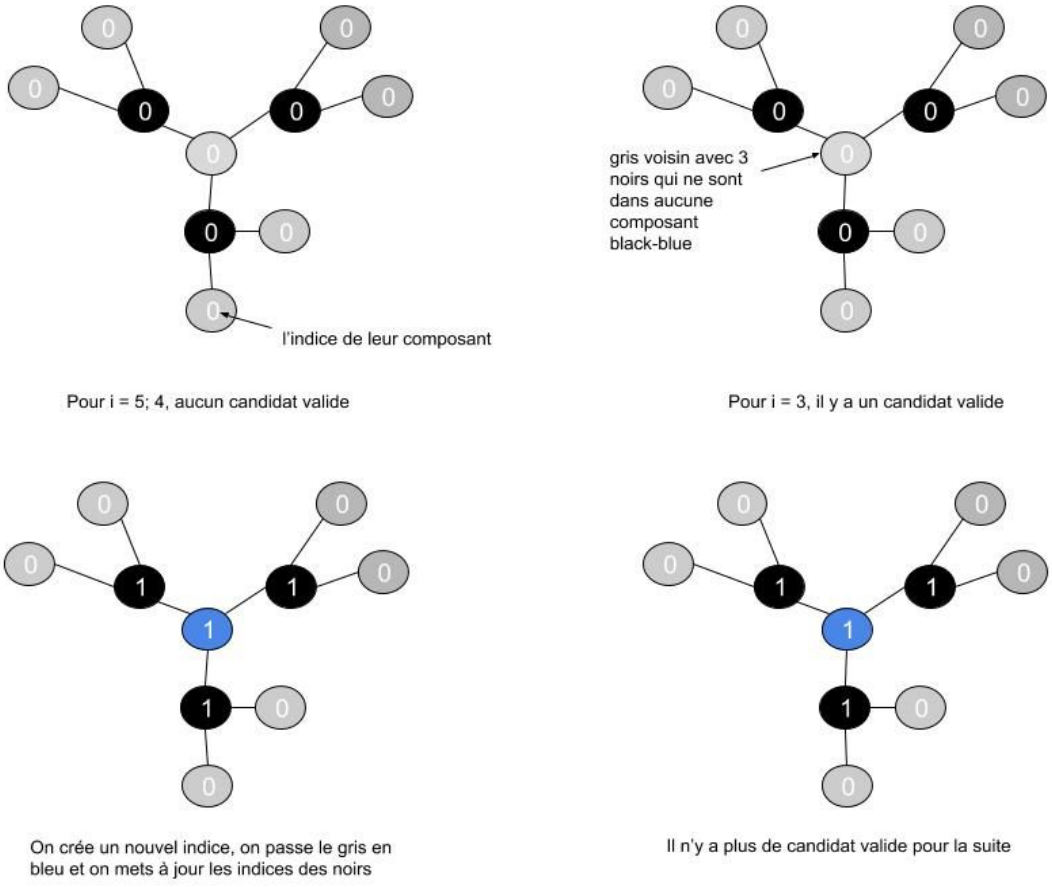

Figure 2 : Schéma du déroulement de l'algorithme qui trouve les noeuds de Steiner

Nous avons donc un ensemble dominant connexe minimum de taille 4. Si on était tombé sur l'ensemble dominant stable de taille 7 on aurait eu un ensemble dominant connexe de taille 10.

Deuxième méthode de construction d'un ensemble dominant connexe

Le but étant de trouver un ensemble dominant connexe le plus petit possible je me suis dit qu'en partant d'un ensemble dominant plus petit que l'ensemble stable dans l'algorithme de Li et al nous obtiendrons facilement un ensemble dominant connexe plus petit que ceux obtenus avec la première méthode.

En utilisant une méthode qui trouve un ensemble dominant plus petit que l'ensemble stable j'obtiens de meilleurs résultats contre un coût en temps un peu plus élevé.

J'utilise une méthode de recherche d'un ensemble dominant naïve :

Tant qu'il reste des noeuds dans mon graphe je cherche le noeud de plus grand degré je l'ajoute à ma solution puis je l'enlève lui et ses voisins de mon graphe.

En suivant cet algorithme nous obtenons un ensemble dominant plus petit que l'ensemble stable mais nous perdons la propriété de stabilité.

Ensuite en utilisant l'algorithme des noeuds de Steiner sur cet ensemble nous obtenons un ensemble dominant connexe plus petits que celui que nous obtenons avec la méthode Li et Al.

Résultats

Méthodes de validation

Pour valider mes résultats j'ai implémenté différentes méthodes, il y a une méthode qui me dit si un ensemble de noeuds est connexe, si un sous ensemble de noeuds est dominant et si un sous ensemble de noeuds est un ensemble stable.

La distance utilisé dans ses algorithmes est la distance euclidienne.

isConnected

C'est une méthode qui prend un ensemble de noeuds qui applique l'algorithme de Floyd-Warshall sur cet ensemble et qui me donne la matrice de distance et de chemin entre chaque noeud.

Ensuite pour tous les noeuds je vérifie qu'ils ont un chemin vers tous les autres noeuds, si c'est le cas l'ensemble de noeuds est connexe sinon il ne l'est pas.

isMIS

Cette méthode prend un ensemble de noeuds et un sous ensemble de cet ensemble.

Pour chaque noeud de mon sous ensemble (mon potentiel MIS) je vérifie que dans ses voisins il n'y a aucun noeuds du sous ensemble.

Ensuite je vérifie dans les voisins des voisins de ce noeuds si il y a au moins un noeud du sous ensemble si c'est le cas je passe au noeud suivant du sous ensemble.

Quand j'ai vérifié ces propriétés pour tous les noeuds du sous ensemble c'est que mon sous ensemble est stable.

isValid

Cette méthode prend un ensemble de noeuds et un sous ensemble de cet ensemble.

Pour chaque noeud du sous ensemble je le retire de mon ensemble ainsi que ses voisins et si a la fin mon ensemble de noeuds est vide c'est que mon sous ensemble domine tous les noeuds de mon ensemble donc c'est un ensemble dominant.

Graphes géométriques aléatoires

Pour générer les graphes je me contente de générer aléatoirement des coordonnées x et y de façon à respecter les dimensions que je me suis fixé en largeur et en hauteur. Comme ses dimensions ne sont pas énormes j'ai très souvent des graphes connexes (avec un seuil de 55).

J'ai donc fait plusieurs graphes dans ma base de données et mes tests sont faits sur ces graphes:

- **Dimensions** : 1200 x 1000 (Largeur x Hauteur)
- **Seuil** : 55 (distance euclidienne entre deux noeuds pour qu'il y ai une arête)
- graphe 1 : 1000 noeuds
- graphe 2 : 2000 noeuds
- graphe 3 : 3000 noeuds
- graphe 4 : 4000 noeuds
- graphe 5 : 5000 noeuds

Mesures

En utilisant la première méthode pour générer les ensembles stables :

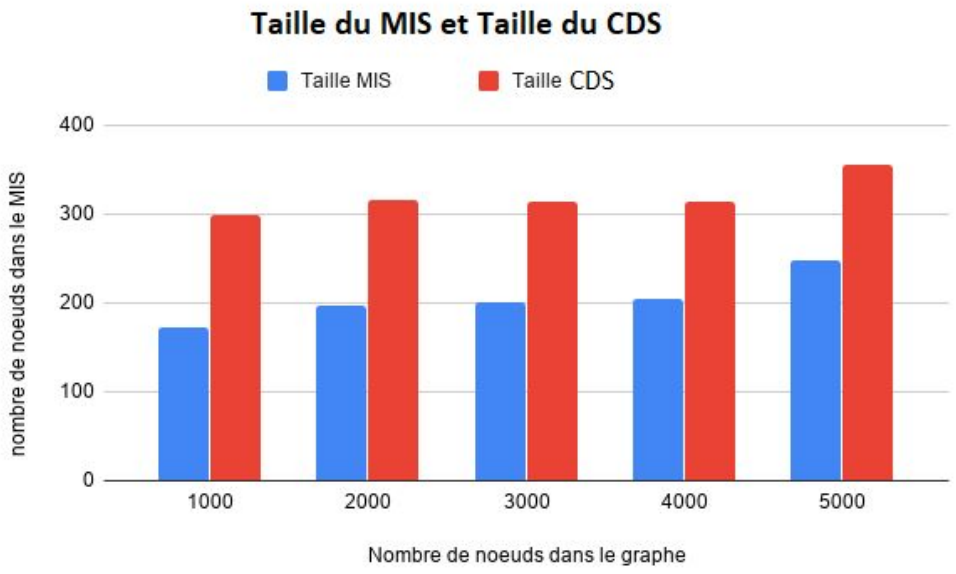

Figure 3 : Taille des ensembles stables et ensembles dominants connexes méthode Li et Al

En utilisant la méthode pour générer les ensembles dominants de manière naïve :

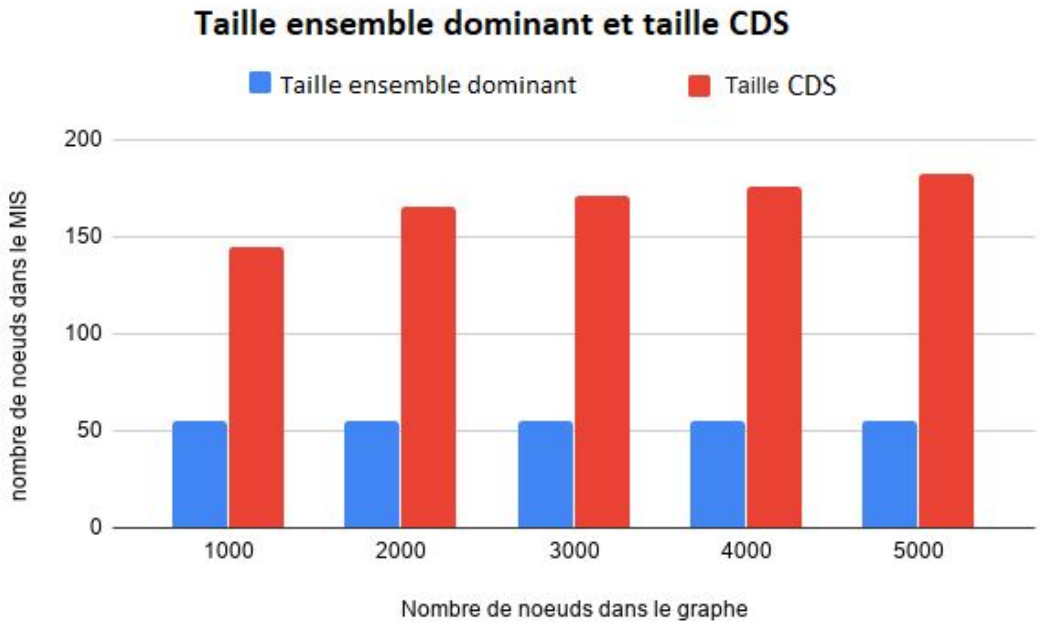

Figure 4 : Taille des ensembles dominants et ensembles dominants connexes méthode naïve

On remarque qu'avec la deuxième méthode sur les mêmes graphes le CDS que nous trouvons est bien plus petit qu'avec la première méthode.

Mais la deuxième méthode ajoute bien plus de noeud de Steiner que la première.

En utilisant la première méthode pour générer les ensembles stables :

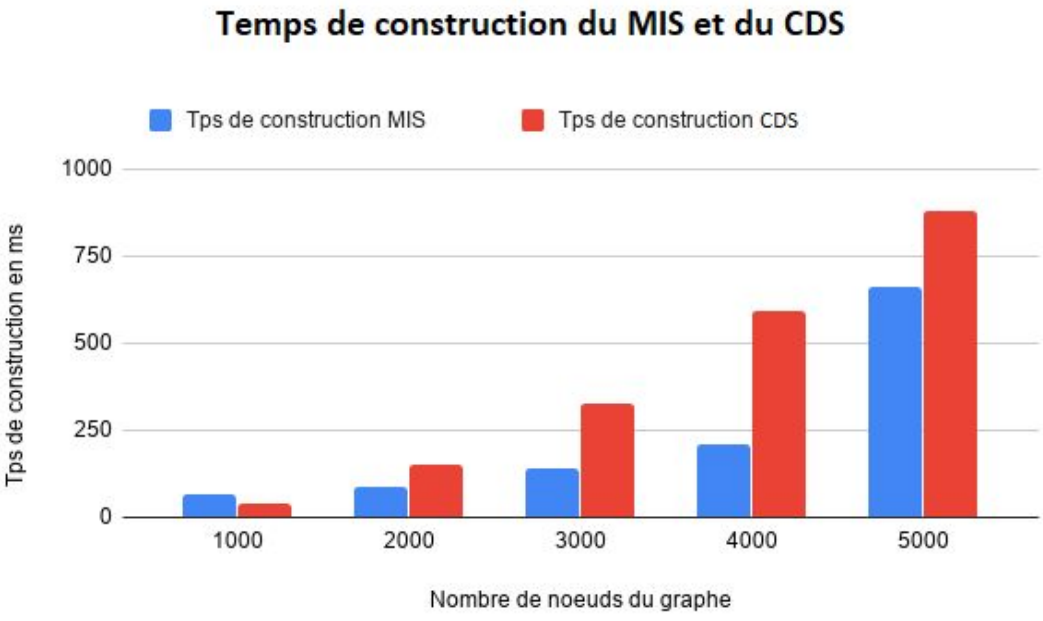

Figure 5 : Temps de construction des ensembles stables et ensembles dominants connexes méthode Li et al

En utilisant la seconde méthode :

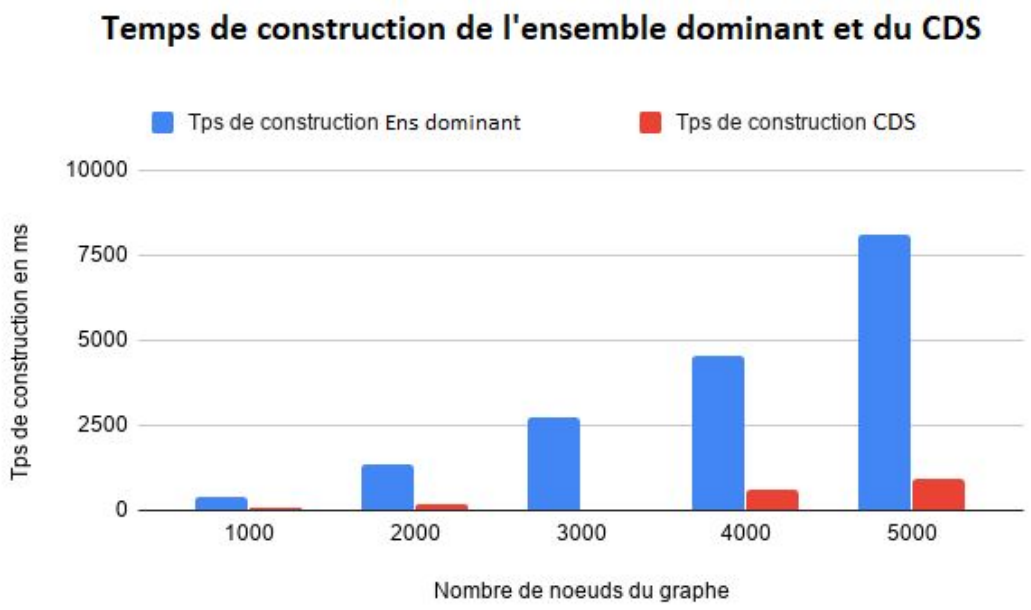

Figure 6 : Temps de construction des ensembles dominants et ensemble dominants connexes méthode naïve

Ici on voit bien que calculer un ensemble dominant de manière naïve est bien plus coûteux que de chercher un ensemble stable avec la méthode décrite plus haut.

Pour le prochain graphique j'ai fait varier le seuil sur le graphe de 5000 noeuds afin d'obtenir des graphes qui ont des chemins plus long entre les noeuds.

On peut comparer la taille des ensembles connexes obtenus en fonction des méthodes utilisés :

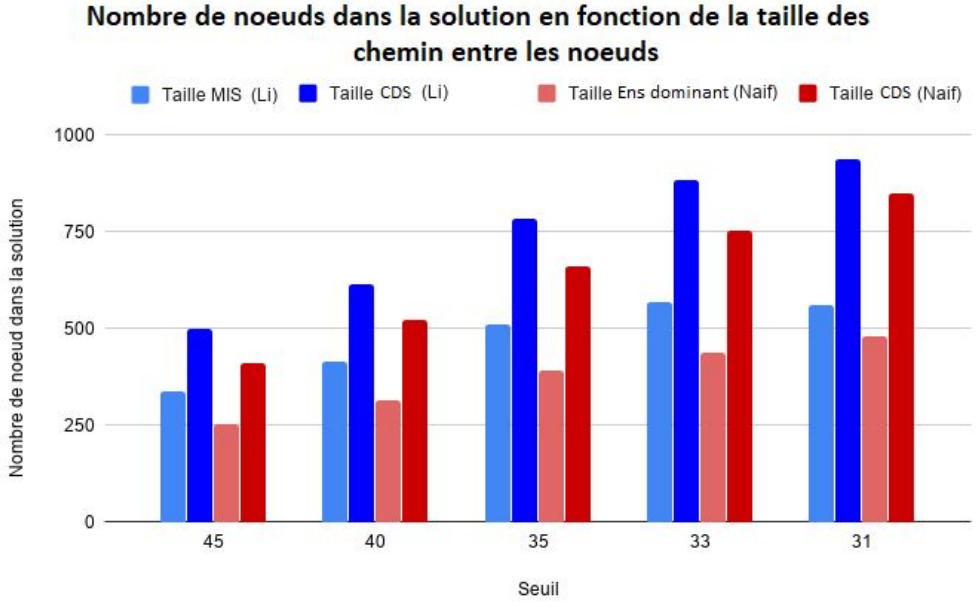

Figure 7 : Taille des ensembles dominants et ensembles dominants connexes obtenus en variant le seuil

Je n’ai pas mis les mesures de temps sur ces mesures car le temps d'exécution de la méthode naïve était beaucoup plus grand que la méthode qui cherche des ensembles stables et que cela rendait les comparaisons peu pertinentes.

Discussion et Conclusion

On peut voir grâce à ces mesures que l'implémentation de l’ensemble dominant stable nous amenés à trouver un ensemble dominant connexe mais qu’il était possible d’améliorer la taille de cette ensemble dominants connexe en partant d’un ensemble dominant plus petit.

Pour l’algorithme de Li et al on aurait donc pu penser à implémenter une forme d’optimisation pour trouver de meilleurs ensembles stables (plus petit) pour améliorer la taille des solutions que nous obtenons tout en ajoutant le moins de noeud de Steiner à notre solution.

Références

- “*On greedy construction of connected dominating sets in wireless networks*” - Yingshu Li, My T. Thai, Feng Wang, Chih-Wei Yi, Peng-Jun Wan et Ding-Zhu Du
- “*Connected domination in ad hoc wireless networks*” - Cadei M, Cheng MX, Cheng X, Du D-Z.