Projet:Ensemble dominants minimaux d'un graphe

Jingwei Zuo

5 December 2015

I. Explication du problème

Soit $G = (V, E)$ un graphe non orienté et connexe. Un sous-ensemble S de sommets de G est dit ensemble dominant de G si tout sommet $v \in V$ est soit un élément de S soit il existe un somment de S adjacent à v.

La taille d'un ensemble dominant est sa cardinalité, c.-à-d., le nombre de sommets dont il est composé. L'ensemble dominant de cardinalité minimale est qualifié d'ensemble dominant minimum.

C'est possible qu'il existe plusieurs ensemble dominant minimum pour un graphe.

Ce que l'on va faire, c'est de poser une solution pour trouver tous les ensembles dominants minimaux d'un certain graphe ou d'un graphe aléatoire, on va afficher leur taille en même temps.

II. Solution algorithmique proposée

L'algorithme énumération:

Entrée: graph G= (V, E)

Sortie: tous les ensembles dominants minimaux et leur taille

initialisation():

```
Debut
        pour k=1 à N faire
                visites[k] <- false ;
//Au debut du programme,tous les sommets du graph n'ont pas visité
        Fin pour;
Fin;
```

émuset(j):

```
Debut
    pour k=1 à N faire
        Si non visites[k] alors
            visites[k] <- true ;
            ensemble E += sommet(k)
            list.add(E);
            émuset(j+1);
            visites[k] <- false ;
        Fin Si
    Fin pour
Fin//cette algorithme va énumérer tous les sous-ensembles d'un ensemble.
```

Maintenant, ce que l'on va faire, c'est de poser une algorithme de trouver les ensembles dominant dans tous les sous-ensembles.
Voici c'est un méthode général:

Donné: un sou-ensemble A, graph G= (V, E)

But: Justifier que s'il est un ensemble dominant

V: ensemble intégral
Selon le graph G, on peut savoir les liaisons parmi les sommets.
On suppose que :

E_A={ B | B est le sommet qui relie avec les sommets du sou-ensemble A}

C'est évidemment qu'il aura des sommets répétitifs dans E_A, on va les enlever. Après cette optimisation, on obtient un ensemble:

E_A_optimisé = { B | B est le sommet non-répétitifs qui relie avec les sommets du sou-ensemble A}

pseudo code de ce méthode

bool jugement(ensemble A)

```
Debut
    Pour C ∈ V faire
        Si (non C ∈ A) && (non C ∈ E_A_optimisé) faire        //s'il existe un sommet  de V qui ne fait pas partie de A, ni de E_A_optimisé, A n'est pas un ensemble dominant
            flag = 1;        //sou-ensemble A n'est pas un ensemble dominant
            return false;
        Fin Si;
    Fin Pour;
    Si (flag ==0) faire      //sou-ensemble A est un ensemble dominant
        output(); //sortir l'ensemble dominant
        return true;
    Fin Si;
Fin
```

ensemble E_D_M() *//dans ce méthode, on va trouver tous les ensembles dominants de la taille minimale.*

```
E_D: ensemble dominant du G;
S_E_D:tous les ensemble dominant du G;
Debut
	Pour E_D ∈S_E_D faire
		trouver les ensembles dominants “E_D_M” de la taille minimale;
	Fin Pour;
Fin;
```

Optimisation sur cette algorithme:

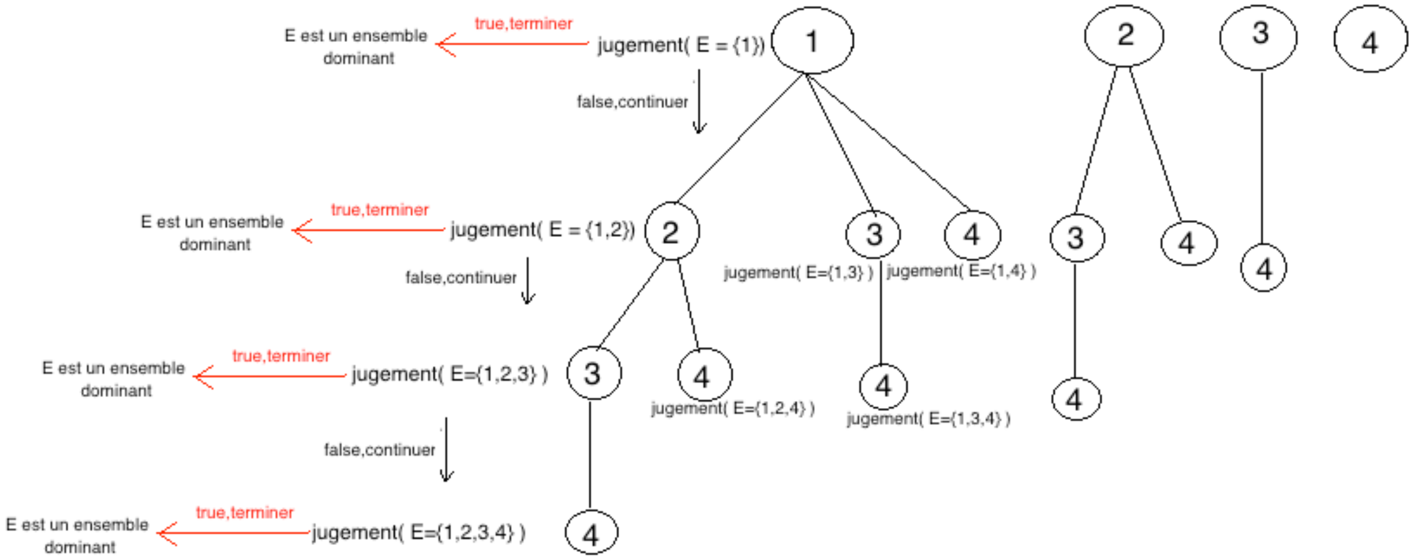
Grace à cette optimisation, on n’a plus besoin d’énumérer tous les sous-ensemble du V;

```
émuset(j);
Debut
	pour k=1 à N faire
		Si non visites[k] alors
			visites[k] <- true ;
			ensemble E += sommet(k)
			Si (jugement(E)) faire
				list.add(E);
				break;
			Fin Si;
			émuset(j+1);
			visites[k] <- false ;
		Fin Si
	Fin pour
Fin
```

Un exemple simple est comme ci-dessous expliquant cette algorithme.

On suppose que N=4

les processus sont dans la figure ci-dessous:



Une fois qu'un sous-ensemble est justifié comme un ensemble dominant, on ne veut plus continuer à lire les sous-ensembles dans ce sens. Et on va revient à l'échelle précédente, et le continuer dans autres sens.