Académie de Montpellier

Université Montpellier II

- sciences et techniques du languedoc -

Diplôme d'études approfondies

- INFORMATIQUE -

Quelques propriétés des graphes distances héréditaires

Guillaume Damiand

Date de soutenance : 27 Juin 1997

Directeurs de stage : Michel Habib

Table des matières

1 Introduction

Un graphe non orienté est distance héréditaire si tous ses chemins sont isométriques. Un chemin d'un graphe est isométrique si la distance entre deux sommets quelconques de ce chemin est la même que la distance dans le graphe.

Une caractérisation des graphes distances héréditaires est donnée et démontrée en [1] sous la forme du théorème 1

Théorème 1 *Les conditions suivantes sont équivalentes :*

1. *G est distance héréditaire*
2. *G n'a pas de sous-graphe isomorphe à un graphe de la figure 1 ou à un* $C_k \quad (k \geq 5)$ *sans corde*
3. *Tous les* $C_k \quad (k \geq 5)$ *de G ont deux cordes croisées*
4. *Tous les sous-graphes de G ont un sommet pendant ou une paire de sommets jumeaux*

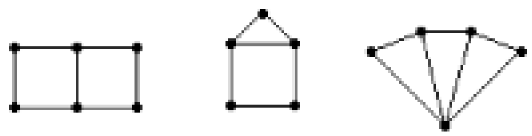

Fig. 1 – Le domino, la maison et le diamant

La propriété 4 du théorème 1 appliquée récursivement permet de détruire entièrement un graphe distance héréditaire en utilisant exclusivement les opérations de contraction de deux sommets jumeaux et de destruction d'un sommet pendant. On appellera élagage d'un graphe sa destruction à l'aide de ces opérations.

Le sujet du stage était le problème ouvert suivant :

Problème 1 *Soit un graphe biparti $G = (X, Y, E)$. Il s'agit de trouver un algorithme linéaire calculant le graphe réduit de ce graphe suivant les deux opérations :*

1. *Destruction d'un sommet pendant*
2. *Contraction de deux sommets jumeaux*

Pour cela, nous avons commencé par nous intérésser aux graphes distances héréditaires pour essayer de voir si l'algorithme de reconnaissance de ces graphes ([1]) ne pouvait pas être utilisé pour ce problème. Nous rendant compte que cet algorithme n'était pas complet, nous avons alors travaillé sur les graphes distances héréditaires, pour dans un premier temps, corriger cet algorithme de reconnaissance.

Nous allons commencer par rappeler quelques définitions et notations au chapitre 2. Puis nous étudierons l'élagage des cographes au chapitre 3, qui nous sera utile pour les deux algorithmes de reconnaissance des graphes distances héréditaires des chapitres 4 et 5. Enfin nous étudierons quelques propriétés intéressantes des graphes distances héréditaires utilisant les arbres de décompositions par substitutions au chapitre 6.

2 Notations et Définitions

Soit $G = (V, E)$ un graphe non orienté. On note n le cardinal de V et m le cardinal de E.

Un *chemin* de G est une séquence $(x_1, x_2, \ldots, x_k)$ de sommets de G tel que $\forall i \in \{1, \ldots, k-1\} : x_i x_{i+1} \in E$. La *distance* entre deux sommets a et b de G notée $d(a, b)$ est la longueur du plus court chemin entre a et b dans G. Un *cycle* est un chemin fermé de G, c'est à dire un chemin tel que $x_1 = x_k$. Une corde d'un cycle est une arête entre deux sommets non consécutif du cycle. Un C_k est un cycle ayant k sommets. On parle de cycle long pour les C_k tel que $k \geq 5$.

$G'(V', E')$ est un *sous-graphe* de G ssi $V' \subseteq V$ et pour tout couple de sommets x et y de V', $xy \in E'$ ssi $xy \in E$. On note $G(Y)$ le sous-graphe de G induit par $Y \subseteq V$.

On note $N(x)$ le voisinage du sommet x dans G. Deux sommets de G, x et y, sont jumeaux ssi $N(x) \setminus \{y\} = N(y) \setminus \{x\}$. On dira que x et y sont de *faux jumeaux* s'ils ne sont pas adjacents, et qu'ils sont de *vrai jumeaux* sinon.

2.1 Séquence d'élagage

On peut maintenant donner la définition de la séquence d'élagage.

Définition 1 *Une* séquence d'élagage *d'un graphe G est une séquence de mots (s_2, s_3, ..., s_n) tel qu'il existe une numérotation des sommets de G de $1, \ldots, n$ vérifiant pour tout $i \in \{2, \ldots, n\}$:*

1. *s_i est un mot de la forme Pj, Fj ou Tj avec $j < i$*
2. *$G(\{1, \ldots, i\})$ s'obtient à partir de $G(\{1, \ldots, i-1\})$ en insérant le sommet numéro i tel qu'il soit respectivement un sommet pendant, un faux ou un vrai jumeau du sommet numéro j.*

La lettre P, F ou T est appelée le type du sommet numéro i, et j est appelé le sommet relatif de i.

Par extension, nous parlerons de séquence d'élagage également pour une séquence de mots (s_2, s_3, ..., s_n) avec chaque mot s_i qui est de la forme xPy, xFy ou xTy, avec cette fois x et y qui sont les sommets eux-mêmes et pas leur numéro. En effet ces deux séquences sont équivalentes, on peut passer facilement de l'une à l'autre.

Quand on a cette variante de séquence d'élagage, on peut très facilement calculer la "vraie" séquence d'élagage, en faisant deux parcours de cette séquence : un pour numéroter les sommets, on affecte le numéro i au sommet x du mot s_i de la forme xPy, xFy ou xTy et le numéro 1 au sommet relatif du mot s_2. Lors du deuxième parcours, on va remplacer chaque mot s_i de la forme xPy, xFy ou xTy, par le mot de la forme Pj, Fj ou Tj avec j qui est le numéro de y.

Dans l'autre sens, si on a une séquence d'élagage et que l'on veut calculer la variante de la séquence d'élagage, il suffit de remplacer chaque mot s_i de la forme Pj, Fj ou Tj par le mot de la forme xPy, xFy ou xTy avec x qui est le sommet ayant pour numéro i, et y qui est le sommet ayant pour numéro j.

Corollaire 1 *Un graphe G a une séquence d'élagage ssi G est distance héréditaire.*

Démonstration :
Sens $\Rightarrow$: Si G a une séquence d'élagage, alors il est distance héréditaire.
Démonstration par l'absurde. On suppose qu'on a un graphe G ayant une séquence d'élagage et que G n'est pas distance héréditaire. Soit i le sommet de la séquence d'élagage tel que $G(\{1, \ldots, i-1\})$ est distance héréditaire, et $G(\{1, \ldots, i\})$ ne l'est plus. (Ce sommet existe forcément, car tout graphe ayant 4 sommets ou moins est distance héréditaire, et $G(\{1, \ldots, n\})$ ne l'est pas par hypothèse.) Alors d'après la propriété 2 du théorème 1, $G(\{1, \ldots, i\})$

a un sous-graphe isomorphe à un graphe de la figure 1 ou à un cycle long sans corde. Le sommet i appartient forcément au sous-graphe "interdit" de $G(\{1,\ldots,i\})$, sinon ce sous-graphe serait déjà dans $G(\{1,\ldots,i-1\})$. Mais pour chacun des 4 graphes "interdits", on peut vérifier qu'il est impossible d'ajouter un sommet de type pendant ou jumeau d'un autre sommet pouvant générer ce graphe interdit. D'ou contradiction avec G n'est pas distance héréditaire.
Sens $\Leftarrow$: Si G est distance héréditaire, alors il a une séquence d'élagage.
La propriété 4 du théorème 1 dit que tout sous-graphe de G a un sommet pendant ou deux sommets jumeaux. On peut donc détruire un sommet pendant ou contracter deux sommets jumeaux du graphe, et insérer en début de la séquence d'élagage l'opération effectuée avec les sommets relatifs. Le graphe ainsi obtenu est un sous-graphe de G, on peut donc appliquer récursivement la même propriété. On va finalement bien obtenir une séquence d'élagage de G. □

2.2 Les niveaux de distances

On appelle niveau de distance L_i de G par rapport à un sommet a, l'ensemble des sommets x de G tel que $d(x,a)=i$. Ces niveaux de distances se calculent par un parcours en largeur d'origine a. Quand on a calculé les niveaux de distances, on note p le plus petit entier tel que $L_{p+1}=\emptyset$ et donc la composante connexe de G contenant a est partitionnée en $[a, L_1, L_2, \ldots, L_p]$. Pour chaque sommet $x \in L_i$, on note N^-(x) (resp. $N^=$ et N^+) le voisinage de x dans L_{i-1} (resp. L_i et L_{i+1}), et d^-(x) (resp. $d^=$ et d^+) le cardinal de N^-(x) (resp. $N^=$ et N^+).

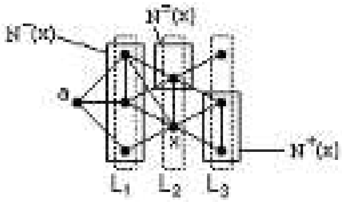

FIG. 2 – Un exemple de graphe avec les niveaux de distances

Ces niveaux de distances nous permettent d'avoir une nouvelle caractérisation des graphes distances héréditaires, qui est démontrée en [2].

Théorème 2 *Soit G un graphe connexe et a un sommet de G. On a calculé les niveaux de distance de G d'origine a. G est distance héréditaire ssi les* *conditions suivantes sont vérifiées, pour tout $i \in \{1, 2, \ldots, p\}$:*

1. *Si x et y sont deux sommets dans la même composante connexe de $G(L_i)$, alors $N^-(x) = N^-(y)$*
2. *$G(L_i)$ est un cographe*
3. *Si $x \in N^-(u)$ et $y \in N^-(u)$ sont dans deux composantes connexes différentes X et Y, alors u est adjacent à tous les sommets de X et de Y, et $N^-(x) = N^-(y)$*
4. *Si x et y sont dans deux composantes connexes différentes de $G(L_i)$, alors $N^-(x)$ et $N^-(y)$ sont disjoints ou un des deux ensemble est inclus dans l'autre.*
5. *Si $x \in N^-(u)$ et $y \in N^-(u)$ sont dans la même composante connexe de $G(L_{i-1})$, alors les sommets de cette composante connexe n'appartenant pas à $N^-(u)$ sont soit adjacents à x et à y, soit à aucun des deux.*

3 Elagage d'un cographe

Les cographes sont les graphes pouvant s'obtenir à partir du graphe réduit à un seul sommet à l'aide des opérations de composition en série et en parallèle. Les cographes sont une sous-classe de la classe des graphes distances héréditaires.

On sait que tout sous-graphe d'un cographe admet deux sommets jumeaux, et donc un cographe est entièrement destructible en appliquant récursivement l'opération de contraction de deux sommets jumeaux.

Il existe plusieurs algorithmes linéaires de reconnaissance des cographes [3, 4], qui retournent, si le graphe est un cographe, un coarbre.

Un coarbre est un arbre associé à un cographe vérifiant les propriétés :

1. Les feuilles de l'arbre correspondent aux sommets du cographe.
2. Chaque noeud interne[1] de l'arbre est de type Série ou Parallèle et a au moins deux fils.
3. Un noeud interne n'est pas du même type que son père.
4. Deux sommets du graphe sont adjacents ssi ils correspondent à deux feuilles de l'arbre ayant comme premier ancêtre commun un noeud de type série.

[1] Un noeud qui n'est pas une feuille de l'arbre

3.1 L'algorithme

On reprend ici l'idée de Peter L. HAMMER et Frédéric MAFFRAY [1] qui est d'utiliser le coarbre pour élaguer un cographe. En effet, on peut voir très facilement que deux sommets d'un cographe sont jumeaux ssi les feuilles correspondantes dans le coarbre ont le même noeud père. Quand on a plusieurs feuilles ayant le même père, on sait que les sommets correspondant du graphe sont jumeaux, et on va donc les contracter en détruisant en fait tous les sommets sauf un (et en notant dans la séquence d'élagage les destructions effectuées). En itérant ces opérations, et sachant que tout sous-graphe d'un cographe est un cographe, on va finalement contracter entièrement le graphe en un seul sommet et produire la séquence d'élagage.

Algorithme 1: Elagage_Cographe

```
Données : G=(V,E) un cographe
Résultat : Construit la séquence d'élagage de G.
début
    T ⟵ le coarbre de G
    D ⟵ la liste des noeuds de T n'ayant que des feuilles pour fils
    tant que D ≠ ∅ faire
        n ⟵ un noeud de D
        x ⟵ un fils de n
        pour Tout les autres fils y de n faire
            si le type de n est Série alors
                Ajouter en début de la séquence d'élagage yTx.
            sinon
                Ajouter en début de la séquence d'élagage yFx.
        Remplacer le noeud n par x
        si le père de n n'a plus que des feuilles pour fils alors
            Ajouter le père de n dans D
fin
```

3.2 Démonstration

Théorème 3 *L'algorithme Elagage_Cographe (G) calcule une séquence d'élagage de G.*

Démonstration : Tous les noeuds de la liste D n'ont que des feuilles pour fils. On sait que cette liste n'est pas vide pour un cographe, car un cographe a forcément au moins deux sommets jumeaux. Quand on prend un noeud n, dans la boucle tant que, on a tous les sommets correspondants aux fils de ce noeud qui sont jumeaux. On prend x un fils de n aléatoirement. On note $x_1, \ldots, x_k$ les autres fils de n ($k \geq 1$ car tout noeud interne d'un coarbre a au moins deux fils). Soit $V' = V \setminus \{x_1, \ldots, x_k\}$. Alors $\forall i \in \{1, \ldots, k\}$, on a $G(V' \cup \{x_1, \ldots, x_i\})$ qui s'obtient à partir de $G(V' \cup \{x_1, \ldots, x_{i-1}\})$ en insérant x_i comme un vrai jumeau de x si le type de n est série, comme un faux jumeau de x sinon.

Une fois traité tous ces x_i, on remplace n par x, le seul sommet restant. On a donc supprimé du coarbre tous les sommets déjà inséré dans la séquence d'élagage. On met à jour la liste D dans le cas ou le père de n n'a plus que des feuilles pour fils. Ici, on a le coarbre qui représente $G(V')$ qui est un sous-graphe de G, donc un cographe. On va donc itérer ce traitement jusqu'à avoir traité tous les sommets. On va donc bien calculer la séquence d'élagage de G. $\square$

Remarque : On va avoir besoin, dans les algorithmes de reconnaissance, de contracter chaque composante connexe d'un cographe en un seul sommet. Cela peut se faire très facilement. En effet, si G a plusieurs composantes connexes, alors on a forcement la racine du coarbre qui est de type parallèle et chaque fils de la racine représente une composante connexe. Donc pour contracter chaque composante connexe, on garde le même algorithme mais on va s'arrêter quand le noeud choisi n est la racine. En effet, dans ce cas, on a déjà contracté toutes les composantes connexes en un seul sommet, sinon n n'aurait pas que des feuilles pour fils.

3.3 Complexité

On détruit le coarbre au fur et à mesure de son traitement, et donc on ne passe jamais plusieurs fois par la même arête de l'arbre. De plus, quand on a fini de traiter un noeud, ce noeud est remplacé par une feuille et ainsi on ne va plus le retrouver dans la liste D. On ne traite jamais deux fois le même noeud. La complexité est donc linéaire en nombre de noeuds et d'arêtes du coarbre, qui est linéaire en nombre de noeuds et d'arêtes du graphe.

4 Un algorithme de reconnaissance en deux passes

Cet algorithme se décompose en deux phases. La première calcule ce qu'on appelle une séquence d'élagage candidate (S.E.C). Si le graphe est distance héréditaire c'est une séquence d'élagage, mais si le graphe n'est pas distance héréditaire, cette séquence ne correspond en fait à rien. Donc la deuxième phase de cet algorithme va être de vérifier la validité de cette séquence d'élagage candidate.

La première phase est en fait quasiment le même algorithme que la procédure de Peter L. HAMMER et Frédéric MAFFRAY [1].

Algorithme 2: Calcule_S.E.C

Données : $G = (V, E)$ un graphe non orienté
Résultat : Une séquence d'élagage candidate de G
début
  $W \longleftarrow V$
  tant que $W \neq \emptyset$ **faire**
    $a \longleftarrow$ un sommet de W
    Calculer les niveaux de distances $L_1, L_2, \ldots, L_p$ à partir de a
    Vérifier que G(L_p) est un cographe
    pour $j \longleftarrow p, p-1, \ldots, 2$ **faire**
      Contracter chaque composante connexe de G(L_j)
      Trier les sommets x de L_j par ordre croissant sur $d^-(x)$
1   Détruire les sommets x de L_j tels que $d^-(x)$=1
      Vérifier que G(L_{j-1}) est un cographe
      pour chaque $x \in L_j$ *pris dans l'ordre croissant sur* $d^-(x)$ **faire**
        Contracter N^-(x) en un seul sommet
2   Détruire x
    Contracter chaque composante connexe de G(L_1)
3   Détruire les sommets restants x de L_1
    $W \longleftarrow W \setminus \{a\}$
fin

Remarques :

1. En 1, 2 et 3, x est un sommet pendant et on peut donc le détruire en insérant en début de la S.E.C, le mot xPy avec y le sommet relatif, c'est à dire le seul sommet adjacent à x.
2. Quand on vérifie si G(L_{j-1}) est un cographe, si c'est vrai alors T reçoit le coarbre, sinon on retourne faux, G n'est pas distance héréditaire.
3. Contracter les composantes connexes de G(L_j) et contracter chaque N^-(x) en un seul sommet va se faire à l'aide du coarbre T préalable-

ment calculé. Il faut avoir un accès direct sur les feuilles associées aux sommets.

Théorème 4 *L'algorithme Calcule_S.E.C (G) calcule une séquence d'élagage de G ssi G est distance héréditaire.*

Démonstration : Un sens de la démonstration est évident. Si G n'est pas distance héréditaire, alors l'algorithme Calcule_S.E.C (G) ne calcule pas une séquence d'élagage, et ce car il n'en existe pas (d'après le corollaire 1). Cela veut dire qu'au moins un mot de la séquence d'élagage candidate va être faux (c'est à dire de type P et ne concernant pas un sommet pendant, ou de type T où F et ne concernant pas deux sommets jumeaux). Par contre, si G est distance héréditaire, l'algorithme utilise en fait chaque point du théorème 2. Au cours de la boucle pour, on est en train de traiter L_i, et on sait qu'il n'y a pas (plus) de sommets dans L_{i+1}. On vérifie que chaque G(L_i) est un cographe. Si ce n'est pas le cas, ce n'est pas la peine de continuer, G n'est pas distance héréditaire. Sinon, on sait que chaque composante connexe a le même voisinage à gauche (propriété 1 du théorème 2) et donc on peut contracter chaque composante connexe en un sommet, les sommets jumeaux dans G(L_i) sont également des sommets jumeaux de G(W). Ici, G(L_i) est un stable. Les sommets de L_i n'ayant qu'un voisin à gauche (tels que $d^-(x) = 1$) sont donc des sommets pendants de G(W). Puis, on prend les $N^-(x)$ dans l'ordre croissant sur $d^-(x)$. Cet ordre nous garanti de prendre les N^- suivant une extension linéaire de l'ordre d'inclusion. On va donc commencer à traiter les N^- inclus dans d'autres. Cela garantie que quand on prend un $N^-(x)$, on sait qu'il a le même voisinage à droite(à l'aide de la propriété 4 du théorème 2). On sait de plus qu'il a également le même voisinage à gauche (propriété 3 et propriété 1 du théorème 2), et au centre (propriété 3 du théorème 2). Donc ce $N^-(x)$ est un module de G(W) et on peut donc l'élaguer. Après avoir contracté ce $N^-(x)$, on a x qui est un sommet pendant et on va donc le détruire. Quand on a traité tous les N^-, on a L_i qui est vide, et on va passer à L_{i-1}. A la fin, il faut finir l'algorithme en contractant les composantes connexes de L_1 (tous les sommets de L_1 ont comme voisinage à gauche a) et détruire les sommets restants q ui sont pendants. La séquence obtenue est bien une séquence d'élagage, chaque opération notée dans celle-ci concerne bien deux sommets jumeaux ou un sommet pendant. □

La complexité de cet algorithme est linéaire. Quand on traite un L_i, on contracte chaque composante connexe, ce qui se fait en linéaire pour $G(L_i)$ (voir chapitre 3). Le tri des sommets se fait également en linéaire pour $G(L_i)$ (car on trie sur des nombres entiers bornés). Enfin, l'élagage des N^- se fait en linéaire pour $G(L_{i-1} \cup L_i)$. (car on parcourt les arêtes entre L_i et L_{i-1}) Finalement, la complexité totale est bien linéaire pour G.

Algorithme 3: Vérifie_Séquence_Elagage

Données : $(s_2,\ s_3,\ \ldots,\ s_n)$ une séquence d'élagage candidate et G le graphe

Résultat : Vrai si $(s_2,\ s_3,\ \ldots,\ s_n)$ est une séquence d'élagage de G, Faux sinon.

début

    pour $j \longleftarrow n, n-1, \ldots, 2$ **faire**

        $xSy \longleftarrow s_j$

        si $S = P$ **alors**

            si $d(x) \neq 1$ **ou** $xy \notin E$ **alors**

                retourner *Faux*

        sinon

            si $N(x) \cap \{x_1, x_2, \ldots, x_{j-1}\} \neq N(y) \cap \{x_1, x_2, \ldots, x_{j-1}\}$ **alors**

                retourner *Faux*

    retourner *Vrai*

fin

Théorème 5 *L'algorithme Vérifie_Séquence_Elagage retourne Vrai ssi la séquence d'élagage candidate est une séquence d'élagage de G.*

Démonstration : C'est évident d'après la définition 1 de la séquence d'élagage.
Si la S.E.C n'est pas une séquence d'élagage, alors il existe forcément un mot de cette séquence qui est "faux", c'est à dire le type est P est x n'est pas un sommet pendant de y, ou les x et y ne sont pas jumeaux.
Sinon, tous les mots de cette séquence sont "vrai", c'est à dire si le type est P, x est effectivement un sommet pendant, et sinon x et y sont bien jumeaux, cette S.E.C est donc bien une séquence d'élagage.

Lors du calcul de la séquence d'élagage, on remplit une table donnant pour chaque sommet x son numéro dans la séquence d'élagage, c'est à dire i tel que le mot $s_i = xSy$ (Le sommet numéro 1 est le sommet restant à la fin de l'élagage). On note x_i le sommet ayant comme numéro i. On a préalablement trié le voisinage de chaque sommet suivant le même ordre, et donc le test $N(x) \cap \{x_1, x_2, \ldots, x_{j-1}\} \neq N(y) \cap \{x_1, x_2, \ldots, x_{j-1}\}$ se fait en temps linéaire, en parcourant le voisinage de x en même temps que celui de y, et comparant uniquement les sommets de ces voisinage ayant comme numéro i ($i < j$) dans la table.

La complexité est linéaire en nombre de noeuds et d'arêtes de G. En effet on parcourt chaque sommet une seule fois, et pour un sommet on parcourt son voisinage. (On parcourt également le voisinage du sommet relatif mais cela n'augmente pas la complexité, car soit les deux voisinages sont égaux et donc la complexité est la même, soit ils ne sont pas égaux et on sort de l'algorithme dès que l'on s'en rend compte.)

Grâce au théorème 4 et au théorème 5, on peut en déduire que les deux algorithmes enchainés donnent bien un algorithme de reconnaissance des graphes distances héréditaires. Si G est distance héréditaire, Calcule_S.E.C va retourner une séquence d'élagage, et Vérifie_Séquence_Elagage de cette séquence va retourner vrai. Sinon, Calcule_S.E.C va retourner une séquence d'élagage "fausse", et Vérifie_Séquence_Elagage de cette séquence va donc retourner faux.

5 Un algorithme en une seule passe

Contrairement au chapitre précédent, l'algorithme de reconnaissance des graphes distances héréditaires décrit ci-après, le fait en une seule passe. On utilise pour cela le théorème 2. On va tester si le graphe vérifie chacune des propriétés de ce théorème. Si c'est le cas, alors il est distance héréditaire.

5.1 L'algorithme

Algorithme 4: Elagage

Données : $G = (V, E)$ un graphe non orienté ayant n sommets

Résultat : Une séquence d'élagage de G s'il est distance héréditaire, un sous-graphe de G isomorphe à la maison, au domino, au diamant ou à un cycle long sans corde sinon

début

- $W \longleftarrow V$
- **tant que** $W \neq \emptyset$ **faire**
  - $a \longleftarrow$ un sommet de W
  - Calculer les niveaux de distances $L_1, L_2, \ldots, L_p$ à partir de a
  - **pour chaque** *composante connexe A de $G(L_p)$* **faire**
    - Appeler $Test\_Egalit\acute{e}\_A\_Gauche(A)$
  - **si** *$G(L_p)$ n'est pas un cographe* **alors**
    - ECHEC
  - 1 Contracter chaque composante connexe de $G(L_p)$ en un seul sommet
  - **pour** $j \longleftarrow p-1, p-2, \ldots, 1$ **faire**
    - Trier les sommets x de L_{j+1} par ordre décroissant sur $d^-(x)$
    - 2 Détruire les sommets x de L_{j+1} tels que $d^-(x)$=1
    - Appeler $Test\_Arbor\acute{e}e(j)$
    - **pour chaque** $x \in Maximums$ **faire**
      - Appeler $Test\_Egalit\acute{e}\_A\_Gauche(N^-(x))$
    - **si** *$G(L_j)$ n'est pas un cographe* **alors**
      - ECHEC
    - $T \longleftarrow$ Le coarbre de $G(L_j)$
    - Appeler $D\acute{e}truire\_Jumeaux(T)$
    - **pour chaque** *composante connexe A de $G(L_j)$* **faire**
      - Appeler $Test\_Egalit\acute{e}\_A\_Gauche(A)$
    - 3 Contracter chaque composante connexe de $G(L_j)$
  - 4 Détruire les sommets restants de L_1
  - $W \longleftarrow W \setminus \{a\}$

fin

En 2 et 4, les sommets détruits sont des sommets pendants et on va insérer en début de la séquence d'élagage, pour chaque sommet détruit x, le mot xPy avec y le seul sommet relié à x.
Les opérations 1 et 3 vont se faire en appelant Elague_Cographe (voir chapitre 3), et on notera les opérations effectuées dans la séquence d'élagage. En 3, on va appeler cet algorithme sur T, qui est déjà le coarbre de $G(L_j)$.

Algorithme 5: Test_Egalité_A_Gauche

Données : A un ensemble de sommets inclus dans L_j

Résultat : Vérifie si tous les sommets de A ont le même voisinage dans L_{j-1}

début

  $x \longleftarrow$ Un sommet de A

  pour chaque $y \in A \setminus \{x\}$ **faire**

1   **si** $N^-(x) \neq N^-(y)$ **alors** ECHEC

fin

Le test 1 se fait en temps linéaire car on a préalablement trié les N^- de chaque sommet suivant le même ordre.

Algorithme 6: Test_Arborée

Données : j le numéro du niveau de distance en cours de traitement, et le graphe G

Résultat : ECHEC si $\{N^-(x) \mid x \in L_{j+1}\}$ n'est pas une famille arborée de L_j, une partition de L_j en composantes marquées sinon

```
début
    NbCMarquée ⟵ |L_{j+1}|
    Inclus[0] ⟵ 0
    Maximums ⟵ ∅
    pour chaque x ∈ L_j faire
        Marque(x) ⟵ 0
    pour i ⟵ 0 à NbCMarquée faire
        NbSommetsCMarquée[i] ⟵ 0
        CMarquée[i] ⟵ NULL
    NbSommetsCMarquée[0] ⟵ |L_j|
    n ⟵ 1
    pour chaque x ∈ L_{j+1} pris dans l'ordre décroissant sur d^-(x)
    faire
        y ⟵ Le premier élément de N^-(x)
        nactu ⟵ Marque(y)
        si nactu = 0 alors Insérer x à la fin de la liste Maximums
        Représentant[n] ⟵ x
        pour chaque y ∈ N^-(x) faire
            si Marque(y) ≠ nactu alors ECHEC
            Marque(y) ⟵ n
            Décrémenter NbSommetsCMarquée[nactu]
            Incrémenter NbSommetsCMarquée[n]
        Inclus[n] ⟵ nactu
        n ⟵ n + 1
fin
```

Algorithme 7: Détruire_Jumeaux

```
Données : T le coarbre de G(L_j) plus tous les tableaux calculés par
          l'algorithme Test_Arborée
Résultat : Si les N⁻ ne sont pas des modules de G(L_j) alors ECHEC,
           sinon on a contracté chaque N⁻ de L_j et détruit les sommets de L_{j+1}
début
    pour chaque noeud n de T faire
        Sommet(n) ⟵ NULL
    pour i ⟵ NbCMarquée à 1 faire
        pour chaque x ∈ CMarquée[i] faire
            n ⟵ Le noeud de T représentant x
            tant que n est une feuille faire
                n ⟵ Le père de n
                si Sommet(n) ≠ NULL alors
1                   Détruire Sommet(n) de T, de W et de L_j
                    Décrémenter NbSommetsCMarquée[i]
                    si n n'a plus qu'un fils alors
                        Remplacer n par x
                Sommet(n) ⟵ x
        si NbSommetsCMarquée[i] ≠ 1 alors ECHEC
        Sommet(n) ⟵ NULL
        j ⟵ Inclus[i]
        Ajouter x à la liste CMarquée[j]
        Incrémenter NbSommetsCMarquée[j]
2       Détruire Representant[i]
fin
```

En 1, y=Sommet(n) et x sont jumeaux, et donc on va insérer en début de la séquence d'élagage yTx ou yFx suivant s'il existe ou non une arête entre x et y.
Et en 2, y=Représentant[i] est un sommet pendant, et on va donc insérer en début de la séquence d'élagage yPx.

Remarques :

– Dans l'algorithme, chaque fois qu'un sommet est détruit, on le supprime

de W et du L_i dans lequel il se trouve.
- Dans cet algorithme, on sait que G n'est pas distance héréditaire quand on a mis ECHEC comme résultat d'un test. Il reste à retourner un sous-graphe de G isomorphe à un sous-graphe de G isomorphe à la maison, au domino, au diamant ou à un cycle long sans corde. Cela se fait facilement, dans chaque cas, en envisageant tous les motifs possibles. Suivant ces motifs, on saura quel sous-graphe de G on doit retourner.

5.2 Démonstration

5.2.1 Test_Arborée

Les données de cet algorithme sont uniquement le graphe (avec les niveaux de distances) et le niveau de distance en cours de traitement j. On va mettre une marque à chaque sommet de L_j. On dit qu'un sommet est dans la $i^{ème}$ CMarquée s'il a pour marque i. Il y a une CMarquée par sommet de L_{j+1} donc le nombre de CMarquée (NbCMarquée) est égal au nombre de sommets de L_{j+1}.

Au cours de l'algorithme, on a :
- CMarquée[i] contient l'ensemble des sommets étant dans la $i^{ème}$ CMarquée.
- NbSommetsCMarquée[i] est le nombre de sommets appartenant à la $i^{ème}$ CMarquée.
- Maximums est la liste des sommets de L_{j+1} tel que N^{-}(x) est maximum au sens de l'inclusion.
- Inclus[i] donne le numéro de la CMarquée contenant la CMarquée numéro i. (0 si aucune CMarquée ne contient cette CMarquée)
- Représentant[i] est le sommet de L_{j+1} associé à cette CMarquée.

Comme les sommets de L_{j+1} sont triés, on va noter y_i le $i^{ème}$ sommet pour cet ordre. Avec cette notation on a Représentant[i] $= y_i$ et dans la boucle pour, au $i^{ème}$ tour on est en train de traiter y_i.

Comme on traite les sommets de L_{j+1} par ordre décroissant sur d^{-}(x), on sait que si N^{-}(x) $\subseteq$ N^{-}(y), alors on va traiter y avant x. (On commence à traiter les N^{-} ayant le plus de sommets). Et comme quand on traite un sommet x, on renumérote tous les sommets de N^{-}(x) avec n (plus grand que toutes les marques déjà existantes), alors un sommet a toujours comme marque le plus grand i tel que x $\in$ $N^{-}(y_i)$.

Au début, tous les sommets de L_j ont pour marque 0. Donc le nombre NbSommetsCMarquée[0] est égal au cardinal de L_j. Les autres CMarquées n'ont, pour l'instant, aucun sommet.

Invariant 1 *A la fin de l'étape n de la boucle pour chaque* $x \in L_{j+1}$*, on a déjà traité les sommets* $y_1, y_2, \ldots, y_n$ *de* L_{j+1} *: on a* $\{N^-(y_i) \mid i \in \{1, 2, \ldots, n\}\}$ *est une famille arborée de* L_j*.*

Démonstration par induction :
Initialisation Vrai après avoir traité un seul sommet x de L_{j+1}.
Hypothèse On suppose que c'est vrai à l'étape n de la boucle, donc on a déjà traité les sommets $y_1, y_2, \ldots, y_n$ de L_{j+1}.
Induction On est à l'étape $n+1$ et on va donc traiter y_{n+1} que l'on appellera x. Si tous les sommets de N^-(x) ont la même marque nactu, alors N^-(x) $\subseteq$ CMarquée[nactu] $\subseteq N^-(y_{nactu})$ et donc $\forall$ i $\in$ 1, 2, ..., n on a :

- Si $N^-(y_{nactu}) \subseteq N^-(y_i)$ alors N^-(x) $\subseteq N^-(y_i)$
- Si $N^-(y_{nactu}) \cap N^-(y_i) = \emptyset$ alors N^-(x) $\cap N^-(y_i) = \emptyset$
- Si $N^-(y_i) \subset N^-(y_{nactu})$ alors on a traité y_{nactu} avant y_i et donc s'il y a des sommets en commun, ils ont pour marque i. Alors, soit :
  1. $N^-(y_i) \subseteq N^-$(x) impossible car sinon vu l'ordre sur d^-, on aurait dû traiter x avant de traiter y_i ce qui n'est pas le cas.
  2. N^-(x) $\subseteq N^-(y_i)$ impossible car sinon tous les sommets de N^-(x) auraient eu comme marque i alors qu'ils ont comme marque nactu.
  3. N^-(x) $\cap N^-(y_i) \neq \emptyset$ (sans avoir un inclus dans l'autre) impossible car sinon les sommets de l'intersection auraient eu comme marque i, alors qu'ils ont tous pour marque nactu.

  Donc on a forcement N^-(x) $\cap N^-(y_i) = \emptyset$.

Donc dans tous les cas, on a $\{N^-(y_i) \mid$ i $\in \{1, 2, \ldots,$ n+1$\}\}$ est une famille arborée de L_j.

Reste à prouver que si tous les sommets de N^-(x) n'ont pas la même marque, alors $\{N^-(y_i) \mid$ i $\in \{1, 2, \ldots,$ n+1$\}\}$ n'est pas une famille arborée de L_j. Comme tous les sommets n'ont pas la même marque, certains sommets ont donc pour marque n1 et certains ont pour marque n2 (n1 $\neq$ n2). Or N^-(x) ne peut pas contenir entièrement $N^-(y_{n1})$ ni $N^-(y_{n2})$ car sinon on aurait du traiter x avant y1 et y2. Donc on a N^-(x) qui est à cheval sur au moins deux N^- et donc $\{N^-(y_i) \mid$ i $\in \{1, 2, \ldots,$ n+1$\}\}$ n'est pas une famille arborée de L_j, d'ou ECHEC. $\square$

Remarques :

- Si tous les sommets de N^-(x) ont comme marque initiale nactu=0, alors ce N^- n'est inclus dans aucun autre N^-, et donc il est maximal au sens de l'inclusion. On ajoute donc ce sommet à la liste Maximums.
- A la fin de l'algorithme, on a $N^-(y_i)$ qui est égal à CMarquée[i] union toutes les CMarquée[j] tel que Inclus[j]=i, union toutes les CMarquée[k]

tel que Inclus[k]=j, union .... Démo par induction : Vrai juste après avoir traité y_i car il n'existe pas de CMarquée[k] tel que Inclus[k]=i, et donc $N^{-}(y_i)$=CMarquée[i]. Si c'est vrai après avoir traité y_k, alors après avoir traité y_{k+1} si on a enlevé des sommets de CMarquée[i] ou d'une CMarquée[j] tel que Inclus[Inclus[...[j]...] = i, on les a mis dans CMarquée[k+1], et Inclus[k+1] est égal à i (ou à j), et donc l'invariant reste vrai.

5.2.2 Détruire_Jumeaux

On a en paramètre de cet algorithme, le coarbre T du niveau de distance en cours de traitement L_j, ainsi que tous les tableaux calculés par l'algorithme Test_Arborée. On a en chaque noeud n de T un champ Sommet(n) initialisé à NULL.

Invariant 2 *A la fin de l'étape k de la deuxième boucle pour, on a déjà traité les sommets y_1, y_2, ..., y_k de la $i^{ème}$ CMarquée, on a : $\forall$ n $\in$ T, Si Sommet(n) = x $\neq$ NULL alors x est le seul sommet fils de n déjà traité, sinon n n'a pas de fils déjà traité.*

Démonstration par induction :
Initialisation Vrai avant de rentrer dans la boucle, car sommet(n) = NULL pour tout noeud n, et on n'a encore traité aucun sommet.
Hypothèse On suppose que c'est vrai à l'étape k de la boucle, donc on a déjà traité les sommets y_1, y_2, ..., y_k.
Induction On est à l'étape $k+1$ et on va donc traiter y_{k+1} que l'on appellera x. n est la feuille de T représentant x, et donc on va entrer dans la boucle Tant que. Au cours de cette boucle, on remonte dans le coarbre. Si on tombe sur un noeud n pour lequel Sommet(n) = y $\neq$ NULL, alors y est le seul sommet fils de n déjà traité (par hypothèse d'induction). Dans ce cas, on détruit y, et on affecte x à Sommet(n), qui est donc bien le seul sommet fils de n déjà traité. Si après cette destruction, n n'a plus qu'un seul fils, alors c'est forcément x et on change n en x (ce qui ne modifie pas le graphe), ce qui va entraîner une itération supplémentaire car n est maintenant une feuille. Par contre, si le noeud n est tel que Sommet(n) = NULL, alors on n'a traité encore aucun sommet fils de n (par hypothèse d'induction), et donc on affecte x à Sommet(n) qui est ici aussi le seul sommet fils de n déjà traité. Pour les autres noeuds, l'invariant est conservé car on ne les modifie pas. □

Invariant 3 *A la fin de l'étape i de la première boucle pour, on a contracté tous les sommets de la $i^{ème}$ CMarquée en un seul sommet ssi CMarquée[i] était un module de G(W).*

Démonstration en utilisant l'invariant 2 :
On sait qu'après être sortie de la deuxième boucle pour, on a pour tout noeud n du coarbre, si Sommet(n) = x $\neq$ NULL, alors x est le seul sommet (de CMarquée[i]) fils de n.

Sens $\Rightarrow$: Si on a contracté tous les sommets de cette CMarquée en un seul sommet, alors il existe un seul noeud n dans le coarbre pour lequel Sommet(n) $\neq$ NULL. (S'il en existe plusieurs, il y a forcement plus d'un seul sommet dans CMarquée[i]) et cela veut dire que les sommets de cette CMarquée formaient des sous-arbres complets de ce noeud, et donc que cette CMarquée était un module.

Sens $\Leftarrow$: Si cette CMarquée est un module, alors on sait qu'il existe un noeud de T tel que tous les sommets de cette CMarquée forment des sous-arbres complets de ce noeud. Ce noeud contient donc tous les sommets de cette CMarquée, et donc il ne peut pas y avoir d'autres noeuds du coarbre pour lequel Sommet(n) $\neq$ NULL. Donc on a forcément contracté tous les sommets de cette CMarquée en un seul sommet qui est Sommet(n). $\square$
Même démo du sens $\Leftarrow$ mais par contraposée : S'il y a plus d'un noeud pour lequel Sommet(n) $\neq$ NULL, alors ces noeuds ont chacun un seul sommet de cette CMarquée en fils (par Invariant 2), mais ont également au moins un autre fils (car on ne laisse pas dans l'arbre des noeuds n'ayant qu'un seul fils), et donc cette CMarquée n'est pas un module. $\square$

Remarques :
- Si CMarquée[i] est un module, il existe un seul noeud n de T pour lequel Sommet(n) $\neq$ NULL. C'est le dernier noeud sur lequel on s'est arrêté. Donc pour enlever "les marques" (en fait la marque) pour le traitement de la prochaine CMarquée, il suffit de remettre Sommet(n) à NULL pour ce noeud.
- On sait que Représentant[i] est un sommet pendant, car il n'est relié à aucune autre CMarquée (du à l'ordre pris sur les CMarquée qui est l'ordre inverse de l'ordre d'inclusion) et que cette CMarquée a été contracté en un seul sommet. Donc on peut le détruire.
- On fait passer le seul sommet restant de CMarquée[i] dans la composante marquée contenant cette CMarquée (c'est la CMarquée numéro j = Inclus[i]). En effet, on a $N^{-}(y_j)$ qui contient CMarquée[i], et donc ce sommet appartient également à cette CMarquée. (cf. Remarque 2 du chapitre précédent)

Autre démonstration :
On n'a pas besoin de prouver que les CMarquées sont des modules, en fait il

suffit de prouver qu'on contracte bien les sommets jumeaux.

Invariant 4 *A la fin de l'étape k de la deuxième boucle pour, on a déjà traité les sommets y_1, y_2, ..., y_k de la $i^{ème}$ CMarquée, on a : Il n'existe pas de sommets jumeaux non détruits dans $\{ y_1, y_2, \ldots, y_k \}$.*

Démonstration par induction :
Initialisation Vrai au début après avoir traité uniquement un sommet.
Hypothèse On suppose que c'est vrai à l'étape k de la boucle, donc on a déjà traité les sommets y_1, y_2, ..., y_k.
Induction On est à l'étape $k+1$ et on va donc traiter y_{k+1} que l'on appellera x. n est la feuille de T représentant x, et donc on va entrer dans la boucle Tant que. Au cours de cette boucle, on remonte dans le coarbre. Si on tombe sur un noeud n pour lequel Sommet(n) = y $\neq$ NULL, alors y est le seul sommet fils de n appartenant à $\{ y_1, y_2, \ldots, y_k \}$ (cf. Invariant 2). Dans ce cas, x et y sont jumeaux et on détruit y. On itère tant que n est une feuille, donc on va détruire des sommets jumeaux de x. Quand on sort de la boucle Tant que, on a Sommet(n) = x qui est le seul sommet fils de n appartenant à $\{ y_1, y_2, \ldots, y_{k+1} \}$. Donc il ne peut pas y avoir de sommet non détruit de $\{ y_1, y_2, \ldots, y_k \}$ jumeaux avec x, car il faudrait pour cela qu'il soit également fils de n. □

Grâce à cet invariant, quand on sort de cette boucle Pour, on a contracté tous les sommets jumeaux de CMarquée[i]. Si le nombre de sommets restant dans cette CMarquée est supérieur à 1, alors c'est qu'il y a au moins deux sommets non jumeaux dans cette CMarquée. Dans ce cas, on sait qu'on ne pourra pas contracter le graphe en un seul sommet et donc que ce n'est pas un graphe distance héréditaire. (on peut également trouver un sous-graphe interdit en regardant tous les cas possibles) Sinon, on a contracté tous les sommets de cette CMarquée en un seul, on peut donc détruire le représentant de cette CMarquée (qui est pendant) et continuer à traiter les autres Cmarquées.

5.2.3 Elagage

On va tracer l'algorithme.

On calcule d'abord les niveaux de distances L_1, L_2, ..., L_p à partir d'un sommet a quelconque. Puis, pour chaque sommet x $\in L_i$ de la composante connexe contenant a, on partage le voisinage de x en trois :

1. N^-(x) qui est le voisinage à gauche de x, c'est à dire le voisinage dans L_{i-1}.
2. $N^=$(x) qui est le voisinage au centre de x, c'est à dire le voisinage dans L_i.

3. N^+(x) qui est le voisinage à droite de x, c'est à dire le voisinage dans L_{i+1}.

La première étape est de vérifier si chaque composante connexe de L_p à le même voisinage à gauche. Si ce n'est pas le cas, alors le graphe n'est pas distance héréditaire, on peut trouver un sous-graphe interdit. Si c'est le cas, on calcule le coarbre de G(L_p). Si G(L_p) contient un P4, alors le graphe n'est pas distance héréditaire, on peut trouver un sous-graphe interdit. Sinon, on contracte chaque composante connexe en un seul sommet, par destruction itérative des sommets jumeaux (appel de Elagage_Cographe). Donc à la fin, L_p est un stable.

Invariant 5 *Au début de la boucle pour j, L_{j+1} est un stable et L_{j+2} est vide.*

Cet invariant est vrai au premier passage dans la boucle, car j vaut p-1 et on a bien L_p est stable et L_{p+1} est vide.

Dans la boucle pour :

On commence à trier les sommets de L_{j+1} par ordre décroissant sur d^-(x). Les sommets tels que d^-(x) = 1 sont des sommets pendants de G(W), et on peut les détruire.

On appelle Test_Arborée qui vérifie que les N^-(x) forment une famille arborée de L_j. Si ce n'est pas le cas alors le graphe n'est pas distance héréditaire, on peut trouver un sous-graphe interdit.

On vérifie ensuite que les N^-(x) maximums au sens de l'inclusion ont le même voisinage à gauche. Si ce n'est pas le cas alors le graphe n'est pas distance héréditaire, on peut trouver un sous-graphe interdit. Si c'est le cas, alors chaque N^-(x) a le même voisinage à gauche car il est inclus dans un N^-(x) ayant le même voisinage à gauche.

On calcule le coarbre de G(L_j). Si G(L_j) contient un P4, alors le graphe n'est pas distance héréditaire.

Sinon, on appelle Détruire_Jumeaux, qui va vérifier que les N^- sont des modules de G(L_j), (et donc vu les tests faits plus haut des modules de G(W)) et contracter chaque N^-(x) en un seul sommet, tout en détruisant les sommets de Lj+1. Ici, on a donc L_{j+1} est vide.

On est ici dans le même état qu'avant de commencer la boucle pour avec L_p, c'est à dire que le niveau de distance en cours de traitement n'a pas de voisin à droite. Il ne nous reste plus qu'à vérifier que les composantes connexes ont le même voisinage à gauche. On n'a pas besoin de vérifier qu'elles sont "P4-free" car cela a déjà été fait quand on a calcule le coarbre de G(L_j). On contracte chaque composante connexe en un seul sommet grâce à la fonction Elagage_Cographe, sur le coarbre restant à la suite de l'opération Détruire_Jumeaux.

On a donc ici, L_j est un stable, et donc on aura bien conservé l'invariant lors du prochain passage dans la boucle

Après avoir fini la boucle pour, on a L_1 est un stable. On sait que ces sommets ont comme unique voisin à gauche a, le sommet de départ, et donc on les détruit, ce sont des sommets pendants.

On a donc contracté la composante connexe contenant a en un seul sommet (a) en détruisant des sommets pendants et contractant des sommets jumeaux uniquement. Donc cette composante connexe était bien distance héréditaire.

Si ça marche pour toutes les composantes connexes, alors le graphe est distance héréditaire.

5.3 Complexité

5.3.1 Test_Arborée

La première boucle pour prend les sommets x de L_{j+1} un par un, et pour chacun de ces sommets, la deuxième boucle pour explore N^-(x). La complexité est donc en O(n1 + m1) si on note n1 et m1 le nombre de sommets et d'arêtes de G($L_j \cup L_{j+1}$).

5.3.2 Détruire_Jumeaux

Pour chaque CMarquée, le nombre de sommets que l'on va traiter est inférieur ou égal au nombre de sommets du N^- correspondant. Donc on peut majorer le nombre de sommets de chaque CMarquée par le cardinal du N^- associé. On va traiter chaque CMarquée une seule fois, et on va parcourir le sous-arbre contenant les sommets de cette CMarquée. On sait que le nombre de noeuds et d'arêtes d'un coarbre est linéaire en fonction du nombre de sommets du graphe, et comme lors de ce parcours on ne passe pas deux fois par la même arête de l'arbre, alors ce parcours est linéaire en nombre de sommets de chaque CMarquée.

Le nombre de sommets d'une CMarquée étant majoré par le nombre de sommets du N^- associé, la somme des parcours pour chaque CMarquée est donc linéaire en nombre de sommets et d'arêtes de G($L_j \cup L_{j+1}$) (car on parcourt les arêtes entre L_j et L_{j+1}).

La complexité est donc aussi en O(n1 + m1).

5.3.3 Elagage

La boucle Tant que traite les composantes connexes les unes après les autres. Donc pour une composante connexe C, on a :

Calculer les niveaux de distances se fait en linéaire par un parcours en largeur O($n_C + m_C$). Calculer les composantes connexes de L_p se fait en linéaire, et Test_Egalité_A_Gauche est linéaire en nombre de sommets de la composante connexe traitée et en nombre d'arêtes entre L_p et L_{p-1}. On ne traite pas deux fois la même composante connexe et donc la boucle pour est linéaire en nombre de sommets et d'arêtes de G($L_p \cup L_{p-1}$). Le calcul du coarbre de G(L_p) est linéaire en O($n_p + m_p$). (cf [3, 4])

Dans la boucle pour à l'étape j : On va noter n1 et m1 le nombre de sommets et d'arêtes de G($L_j \cup L_{j+1}$), et n0 et m0 le nombre de sommets et d'arêtes de G($L_{j-1} \cup L_j$)

Trier les sommets de L_{j+1} se fait en O($\mid L_{j+1} \mid$) (Car on va trier sur des entiers en connaissant la borne inf. et sup.) Test_Arborée se fait en O(n1 + m1) On appelle Test_Egalité_A_Gauche sur les N^- maximums, donc on ne va pas parcourir deux fois les mêmes arêtes, donc cela se fait en O(n1 + m1). (on parcourt les arêtes entre L_j et L_{j+1}) Le calcul du coarbre de G(L_j) se fait en O($n_j + m_j$). (car on a accès directement au voisinage d'un sommet dans L_j) Détruire_Jumeaux se fait en O(n1 + m1) Calculer les composantes connexes de G(L_j) se fait O($n_j + m_j$), et pour chaque composante connexe, on teste l'égalité à gauche d'ou la complexité en O(n0 + m0). (on parcourt les arêtes entre L_{j-1} et L_j) Elagage_Cographe se fait en O(n1 + m1).

En final, on voit bien que dans la boucle pour, on traite trois fois chaque L_j (en complexité) mais la somme est quand même linéaire en O($n_C + m_C$).

6 Propriétés

6.1 Codage

De par sa définition, la séquence d'élagage est un codage des graphes distances héréditaires. En effet, pour chaque graphe distance héréditaire, on peut calculer une séquence d'élagage, et quand on a une séquence d'élagage, on peut retrouver le graphe associé. La séquence d'élagage est la séquence de mot (s_2, s_3, ..., s_n) avec chaque s_i est un mot de style Pj, Fj ou Tj. On code chaque sommet avec $\lceil \log_2 n \rceil$ bits, et le type de chaque mot de la séquence se code sur 2 bits. La taille exacte de ce codage est donc de $(n-1) \times (2 + \lceil \log_2 n \rceil)$ qui est donc en $O(n \log_2 n)$.

Pour le graphe de la figure 3, on a une variante de séquence d'élagage qui

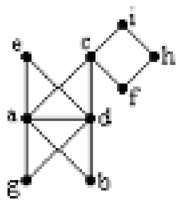

FIG. 3 – Exemple d'un graphe distance héréditaire

est :

$$(\, cPa,\, bPa,\, eFc,\, dTa,\, gFb,\, fPc,\, hPf,\, iFf\,)$$

et la séquence d'élagage associée qui est donc un codage du graphe :

$$(\, P1,\, P1,\, F2,\, T1,\, F3,\, P2,\, P7,\, F7\,)$$

Ce codage n'est pas unique, car la séquence d'élagage ne l'est pas. On a pour ce même graphe d'autres séquences d'élagages, par exemple :

$$(\, P1,\, P1,\, T1,\, F2,\, P2,\, F3,\, P6,\, F6\,)$$

6.2 Nombre de graphes distances héréditaires

Vu que chaque graphe distance héréditaire possède une séquence d'élagage, on a donc le nombre possible de graphes distances héréditaires qui est inférieur au nombre total de séquences d'élagage différentes. (On n'a pas l'égalité car plusieurs séquences d'élagage peuvent représenter le même graphe.)
Pour un graphe à n sommets, la séquence d'élagage a n-1 mots. Chaque mot s_i de cette séquence peut être de trois type différents, et avoir i-1 sommets relatifs possible. On a donc facilement le nombre de séquences d'élagage possible qui est

$$\prod_{i=2}^{n}(3 \times (i-1)) = 3^{n-1} \times (n-1)! = O((n!)^2)$$

6.3 Décomposition par substitution

Nous allons d'abord présenter quelques rappels sur la décomposition par substitution (voir [5]), puis voir ce que cela donne dans le cas des graphes distances héréditaires.

6.3.1 Rappels sur la décomposition par substitution

L'opération de substitution de graphes est le fait de substituer chacun des sommets d'un graphe H (ayant n sommets), par un graphe $G_i = (V_i, E_i)$. On note $G = H_{1,\ldots,n}^{G_1,\ldots,G_n}$ le graphe obtenu par cette substitution. On dit également que H est le graphe quotient de $G = (V, E)$ selon la partition de V en $V_1, \ldots, V_n$. Cette partition est appelée partition de congruence de G.

Chaque graphe G non orienté peut se décomposer, suivant sa structure :

1. Si G est non connexe, on peut appliquer une décomposition parallèle à G.
2. Si $\bar{G}$ est non connexe, on peut appliquer une décomposition série à G.
3. Sinon, on peut appliquer la décomposition premier à G.

Ces trois cas s'excluent mutuellement, on a donc pour un graphe, une seule de ces décomposition qui est applicable. En appliquant ces décomposition récursivement sur un graphe quelconque, on va avoir un arbre de décomposition par substitution vérifiant :

1. Chaque feuille de cet arbre correspond à un sommet du graphe
2. Chaque noeud interne est de type parallèle, série ou premier

Pour conserver toutes les informations du graphe, on associe à chaque noeud premier le graphe quotient de la décomposition. L'ensemble des feuilles d'un sous-arbre quelconque est un module fort de G.

6.3.2 Pour les graphes distances héréditaires

On considère G connexe. On sait que les graphes distances héréditaires ne contiennent pas un sous-graphe isomorphe à un graphe de la figure 1 ou à un cycle long sans corde. Parmi ces graphes, la maison, le domino et les cycles longs sans corde sont indécomposables, et si G contient un sous-graphe isomorphe à un de ces graphes, alors on aura forcément ce sous-graphe qui est un sous-graphe d'un graphe associé à un noeud premier de l'arbre de décomposition par substitution. Et si G contient le diamant, on peut le retrouver également comme un sous-graphe d'un noeud premier, mais aussi s'il existe dans l'arbre de décomposition un noeud premier n qui n'est pas la racine. En effet dans ce cas n à forcément un ancêtre de type série ou premier. Alors, il existe au moins un sommet fils de ce noeud qui va être relié à tous les sommets du P4 contenu dans le noeud premier, ce qui entraîne la présence d'un diamant.

On peut grâce à ces remarques donner une nouvelle caractérisations des graphes distances héréditaires.

Théorème 6 *Soit G un graphe connexe non orienté. G est distance héréditaire ssi son arbre de décomposition par substitution vérifie :*

1. *Le seul noeud pouvant être de type premier est la racine*
2. *Si la racine est de type premier, alors le graphe premier associé à ce noeud est distance héréditaire*

Démonstration :
Sens $\Rightarrow$: Si G est distance héréditaire, alors son arbre de décomposition par substitution vérifie les propriétés 1 et 2.

Si l'arbre de décomposition par substitution de G a un noeud interne n de type premier qui n'est pas la racine, alors comme G est connexe, il existe forcément un noeud ancêtre de n de type série ou premier (G connexe implique que la racine n'est pas de type parallèle), et ce noeud ayant au moins deux fils, il y a au moins un sommet qui va être relié à tous les sommets du P4 contenue dans le graphe premier associé à n. Donc G contient le diamant d'ou contradiction avec G est distance héréditaire. Donc G vérifie la propriété 1.

Si la racine est de type premier et que le graphe associé n'est pas distance héréditaire, alors comme ce graphe est un sous-graphe de G, G n'est également pas distance héréditaire. G vérifie donc forcément la propriété 2.

Sens $\Leftarrow$: Si l'arbre de décomposition par substitution d'un graphe G vérifie les propriétés 1 et 2, alors G est distance héréditaire.

G ne peut pas contenir un sous-graphe isomorphe à la maison, au domino ou à un cycle long sans corde, car ces graphes étant premiers, ils seraient forcément sous-graphe du graphe associé au seul noeud premier éventuel de l'arbre de décomposition (la racine) ce qui n'est pas possible car dans ce cas, ce graphe associé ne serait pas distance héréditaire.

Et G ne peut pas contenir non plus un sous-graphe isomorphe au diamant, car pour cela, il faudrait avoir un noeud premier (qui contient un P4) et un noeud ancêtre de ce noeud ayant un autre fils. Or, le seul noeud éventuellement premier est la racine, donc c'est impossible. De plus, le diamant ne peut pas être un sous-graphe du graphe associé au seul noeud premier éventuel de l'arbre de décomposition sinon ce graphe ne serait pas distance héréditaire. □

Le théorème 6 nous donne immédiatement un algorithme récursif de reconnaissance des graphes distances héréditaires, mais il n'est malheureusement pas linéaire. En effet, du fait de l'appel récursif, on va traiter plusieurs fois les mêmes sommets. Le calcul de l'arbre de décomposition par substitution se fait en $O(n+m)$, et au pire des cas, on va faire n appels récursifs. D'ou la complexité qui est en $O(nm)$.

Algorithme 8: Reconnaissance Recursif

Données : $G = (V, E)$ un graphe non orienté connexe
Résultat : Vrai si G est distance héréditaire, Faux sinon
début
  $T \longleftarrow$l'arbre de décomposition par substitution de G
  pour chaque *noeud n de T différent de la racine* **faire**
    si *Le type de n est premier* **alors**
      retourner *Faux*
  si *Le type de la racine est premier* **alors**
    $H \longleftarrow$ le graphe premier associé à la racine
    si *H n'a pas de sommet pendant* **alors**
      retourner *Faux*
    sinon
      Supprimer les sommets pendants de H
      Appeler $ReconnaissanceRecursif(H)$
  sinon
    retourner *Vrai*
fin

Cet algorithme est très simple. Dans un premier temps, on calcule l'arbre de décomposition par substitution du graphe G. Puis, on vérifie si cet arbre à un noeud premier qui n'est pas la racine. Si c'est le cas, on sait que G n'est pas distance héréditaire grâce au théorème 6. Si la racine de cet arbre n'est pas de type premier, alors on sait que G est distance héréditaire. (c'est en fait un cographe) Sinon, on va détruire les sommets pendants de G (s'il n'y en a pas, alors on sait que G n'est pas distance héréditaire : il n'a ni sommets jumeaux car il est premier, ni sommet pendant) et tester si ce nouveau graphe est distance héréditaire par un appel récursif.

6.4 $\overline{G}$ et G^2

La question est de savoir, quand G est distance héréditaire, ce qu'il en est de $\overline{G}$ et de G^2.

Pour $\overline{G}$, on voit très facilement qu'il n'y a aucun lien avec G.

En effet, pour la figure 4, on a le cas 1 pour lequel G est distance héréditaire, et $\overline{G}$ qui ne l'est pas. Le cas 2 nous montre un cas ou G et $\overline{G}$ sont tous les deux distances héréditaires. Enfin, le cas 3 nous montre un

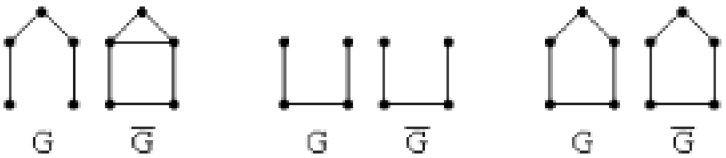

FIG. 4 – Cas 1, 2 et 3

cas ou ni G ni $\overline{G}$ ne sont distances héréditaires.

Et pour G^2, on peut vérifier également que quand G est distance héréditaire, on ne peut rien en déduire pour G^2.

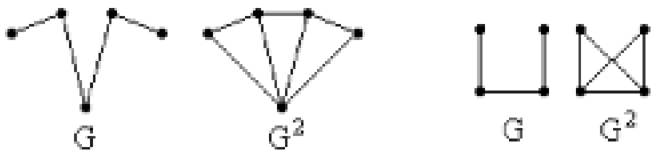

FIG. 5 – Cas 1, 2

En effet, pour la figure 5, on a le cas 1 pour lequel G est distance héréditaire, et G^2 qui ne l'est pas, tandis que le cas 2 nous montre un cas ou G et G^2 sont tous les deux distances héréditaires.

7 L'algorithme de P. Hammer et F. Maffray

Certains de nos algorithmes sont inspirés par l'article de P. Hammer et F. Maffray [1]. Cet article disait donner un algorithme linéaire de reconnaissance des graphes distances héréditaires, mais ce n'était en fait pas le cas : cet algorithme n'était pas complet.

7.1 L'algorithme

Voici la procédure telle qu'elle était proposée par Peter L. HAMMER et Frédéric MAFFRAY dans l'article [1] :

Algorithme 9: PRUNE

Données : $G = (V, E)$ un graphe non orienté ayant n sommets
Résultat : Une séquence d'élagage de G s'il est complètement séparable, un sous-graphe de G isomorphe à la maison, au domino, au diamant ou à un cycle long sans corde sinon

début
- $W \longleftarrow V$
- **tant que** $W \neq \emptyset$ **faire**
  - $a \longleftarrow$ un sommet de W
  - Calculer les niveaux de distances $L_1, L_2, \ldots, L_p$ à partir de a
  - **pour** $j \longleftarrow p, p-1, \ldots, 2$ **faire**
    - Etape 1 : Calculer les composantes connexes de L_j
    - Etape 2 : Pour chaque composante connexe A tel que $|A| \geq 2$ appeler REDUCE(A)
    - Etape 3 : Trier les sommets x de L_j par ordre croissant sur $d^-(x)$
    - Détruire les sommets x de L_j tel que $d^-(x)$=1
    - Etape 4 :Pour chaque x $\in L_j$ pris dans l'ordre croissant sur $d^-(x)$ appeler REDUCE(N^-(x)) et détruire x.
  - $W \longleftarrow W \setminus \{a\}$

fin

Algorithme 10: REDUCE

Données : A un sous-ensemble de V.
Résultat : Si A est un module de G sans P4, on détruit les sommets de A un par un jusqu'à n'avoir plus qu'un seul sommet. Sinon ECHEC.

début
- Etape a : Pour tester que A est un module de G, on prend un sommet x de A et pour chaque sommet z de $A \setminus \{x\}$, on compare la liste N^-(z) avec la liste N^-(x).
- Etape b : Pour tester que G(A) ne contient pas de P4, on appelle un algorithme de reconnaissance des cographes. (voir [3, 4])
- Etape c : On contracte les sommets jumeaux du cographe G(A). (voir chapitre 3)

fin

7.2 Contre-exemples

Malheureusement cette procédure simple n'est pas suffisante pour reconnaître les graphes distances héréditaires. En effet, si le graphe est distance héréditaire, elle donne bien une séquence d'élagage, mais si le graphe n'est pas distance héréditaire, elle va parfois quand même donner une séquence d'élagage, et ce car l'étape a de REDUCE ne teste pas si A est un module de G.

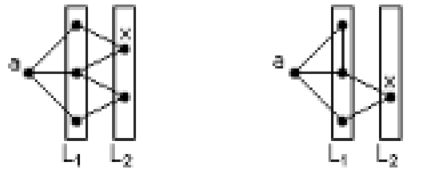

FIG. 6 – Deux contre-exemples : Le domino et la maison

Dans les deux contre-exemples de la figure 6, on a représenté le graphe et les niveaux de distances. Pour ces deux graphes, on a L_2 qui est un stable et il n'a pas de sommet pendant, donc les étapes 1, 2 et 3 de l'algorithme ne vont rien faire. On passe donc directement à l'étape 4, qui prend un x maximum de L_2 pour d^-(x). Dans le premier cas, les deux sommets ont le même d^- et donc on en prend arbitrairement un, et dans le deuxième cas, il n'y a qu'un seul sommet, c'est donc celui qui est choisi. L'algorithme va faire appel à REDUCE(N^-(x)). Dans les deux cas, lors de l'étape a de REDUCE, tous les sommets de ce N^-(x) ont le même voisinage à gauche (le seul voisin à gauche est le sommet a) et l'algorithme en déduit que ce N^-(x) est un module de G, alors qu'il est clair que ce n'est pas le cas. Ces deux cas montrent qu'il faudrait vérifier le voisinage à droite (cas 1) ainsi que le voisinage pour ce L_i (cas 2), c'est à dire en fait le voisinage complet de ce N^-.

8 Conclusion

Dans cette étude de quelques propriétés des graphes distances héréditaires, nous avons principalement donné deux algorithmes linéaires de reconnaissance des graphes distances héréditaires. Ces algorithmes sont importants du fait que le seul algorithme linéaire de reconnaissance existant à notre connaissance [1] n'était pas complet, et que certains algorithmes utilisent comme outil la reconnaissance de ces graphes en linéaire. Nous avons également introduit la notion de séquence d'élagage qui est intéressante, car elle nous donne un codage efficace de ces graphes. L'étude des propriétés de l'arbre de décomposition par substitution nous a permis d'avoir un autre algorithme de reconnaissance, qui bien que non linéaire ait l'avantage d'être très simple. De plus, cet algorithme peut, avec pas beaucoup de modifications, être utilisé pour calculer le graphe réduit d'un graphe quelconque pour les opérations de destruction de sommets pendants et contraction de sommets jumeaux. Cette direction de recherche mériterait d'être approfondi pour essayer de trouver un algorithme linéaire pour le calcul de ce graphe réduit (problème1), problème qui reste aujourd'hui ouvert.

Références

[1] P. Hammer and F. Maffray, Completely separable graphs, Discrete Applied Mathematics 27 (1990) 85-89.

[2] H-J. Bandelt and H.M. Mulder, Distance-hereditary graphs, J. Combin. Theory Ser. B 41 (1986) 182-208.

[3] D.G. Corneil, Y. Perl and L.K. Stewart, A linear recognition algorithm for cographs, SIAM J. Comput. 14 (1985) 926-934.

[4] C. Capelle, A. Cournier and M. Habib, Cograph recognition algorithm revisited and online P4 search, Rapport technique 94073, LIRMM, 1994.

[5] C. Capelle, Décomposition de graphes et permutations factorisantes, Rapport de thèse, Université Montpellier II, Janvier 1997.